Santa Monica Studio™

Preparing AI Systems For GOD OF WAR RAGNARÖK

Göksu Uğur
AI Lead

Santa
Monica
Studio

- Behavior Trees
  - Moving from Lua to Behavior Trees
  - Tools support
  - Key usages of injected behaviors
- Context Actions
  - Achieving modularity & customization
  - Different use cases
- Enhanced Movement and Verticality

Santa Monica Studio

Behavior Trees – it is not the hot new AI technology, so I would like to focus more on our strategy migrating from text based scripting to behavior trees and small wins that made big changes for us

Context actions – How our AI gained awareness about the world around them and how we were able to kitbash smaller modules to achieve more complicated setups

Enhanced mov& verticality – and how these two, combined with some of our other tech enhanced our enemies and combat arenas

Ragnarök is coming ...

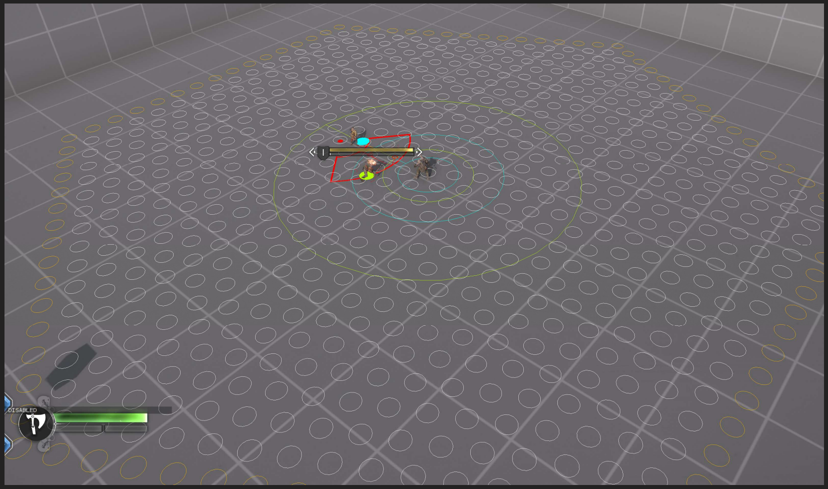
Santa Monica Studio

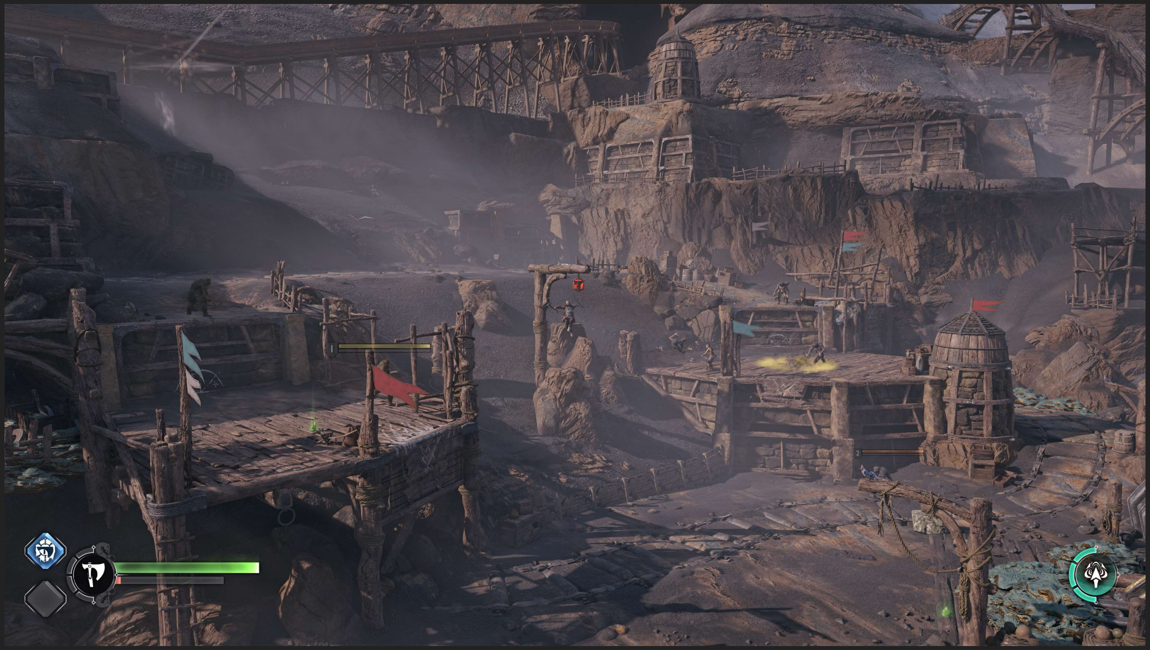
SPOILER ALERT – be warned

So what did the road ahead look like for Ragnarok when we started?
The needs for Ragnarök :
- Very early on we knew we wanted more dynamic and vertical combat arenas
- And to go along with those, we wanted to have enemies that can move through these spaces more freely. And definitely more enemy variety, different shapes, behaviors, movements
- More companions– even though they all use same core systems, they had different styles thus slightly different needs
- Home base spaces that has more AI/NPC going about their business
- More realms and more wildlife

Challenges with existing systems

- Enemy Positioning



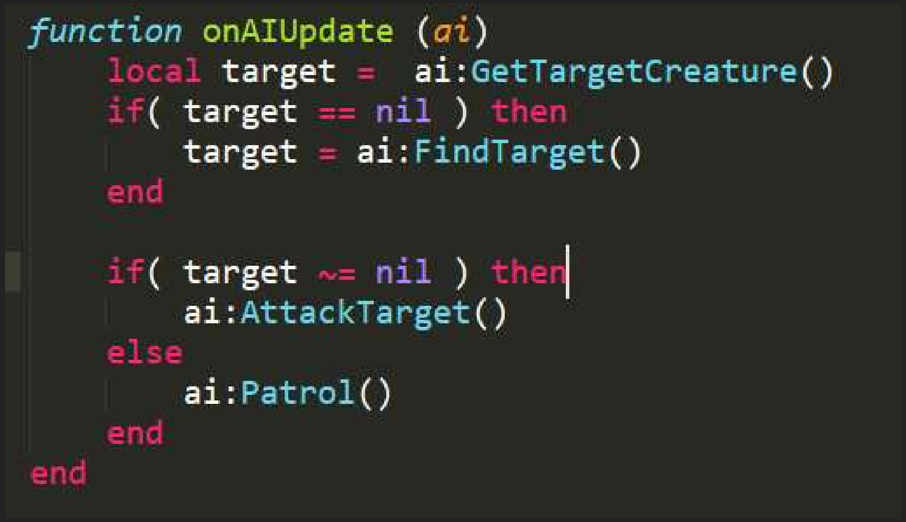


- World annotation
- Companion workflows not expandable
- Lack of centralized decision making for AI
- Using Lua as our scripting language

Santa Monica Studio

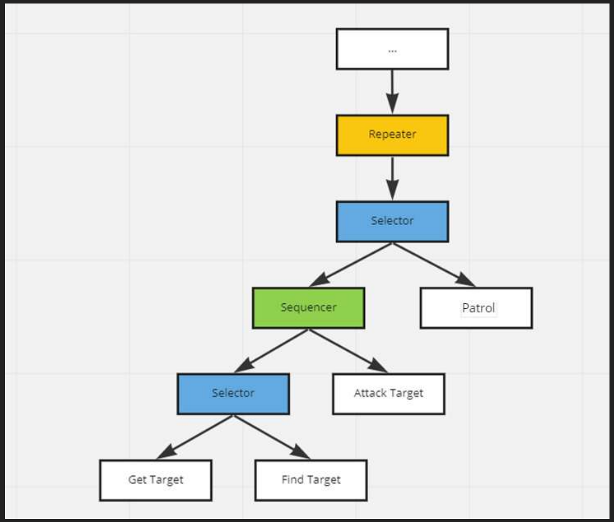
Some of our systems were already long overdue for some refactor even without these new requirements but these sure made us prioritize some over the other. So with that we started taking a more detailed look into the current state of some of these starting with:
-Enemy Positioning: The system itself was holding up fairly well and could handle another game BUT it was very flat and melee-based positioning systems : This is what our positioning looked like, It is just one layer of nodes around player. it had limited capacity to be able to support levels like these: image
-The ways in which we did world annotation for AI behavior was rigid. It needed to be expanded and allow for more systemic behavior as well as easy custom scripted setups
-Companion workflows not allowing for variety : systems around the BOY were implemented very specifically for him. Not modular at all to be able to be expanded and modularized
-Lack of centralize decision making for AI :multiple systems running under the hood that could effect what AI decision making at any given time
-Using Lua for our design scripting language - many challenges that came specifically with this that I will dive into more with detail but we had concerns with: Performance, debugging, Authorization, stability
There were a fair bit of problems we needed to deal with but could not tackle everything with the amount of time we had.So we started with biggest offender and see if we can tackle multiple issues with it

Lua



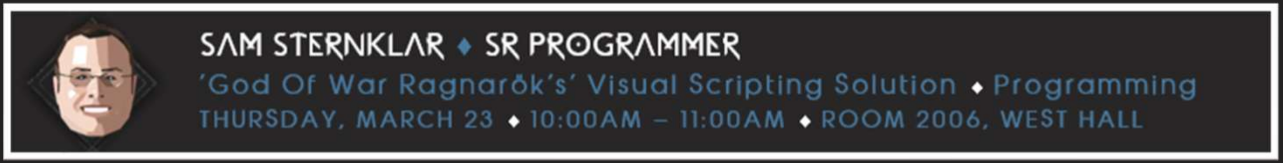
```lua
function onAIUpdate (ai)
    local target =  ai:GetTargetCreature()
    if( target == nil ) then
        target = ai:FindTarget()
    end

    if( target ~= nil ) then
        ai:AttackTarget()
    else
        ai:Patrol()
    end
end
```

Behavior Tree



We decided to move away from LUA ,which was a text based scripting solution we were using for AI behavior and move to using Behavior trees which defines AI behavior as a tree of hierarchical nodes that control the flow

This change would allow use to address Lua specific struggles ,centralize AI decision making and allow us to refactor companion handling and reorganize it to make it easier to implement different companion behaviors with shared functionality

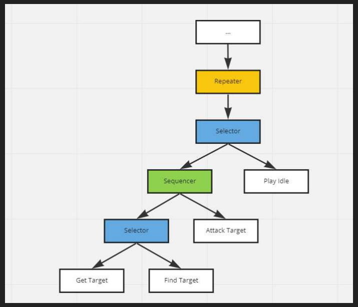


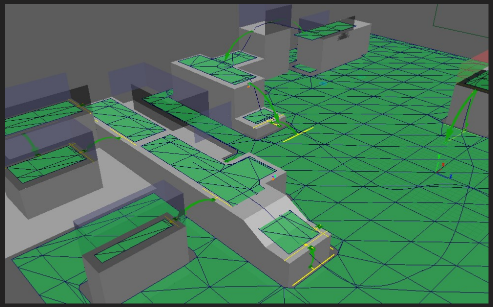
SAM STERNKLAR ♦ SR PROGRAMMER

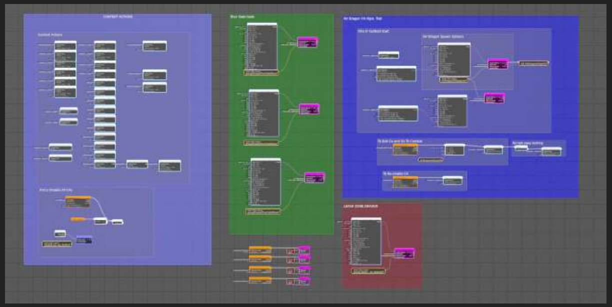
'God Of War Ragnarök's' Visual Scripting Solution ♦ Programming

THURSDAY, MARCH 23 ♦ 10:00AM – 11:00AM ♦ ROOM 2006, WEST HALL

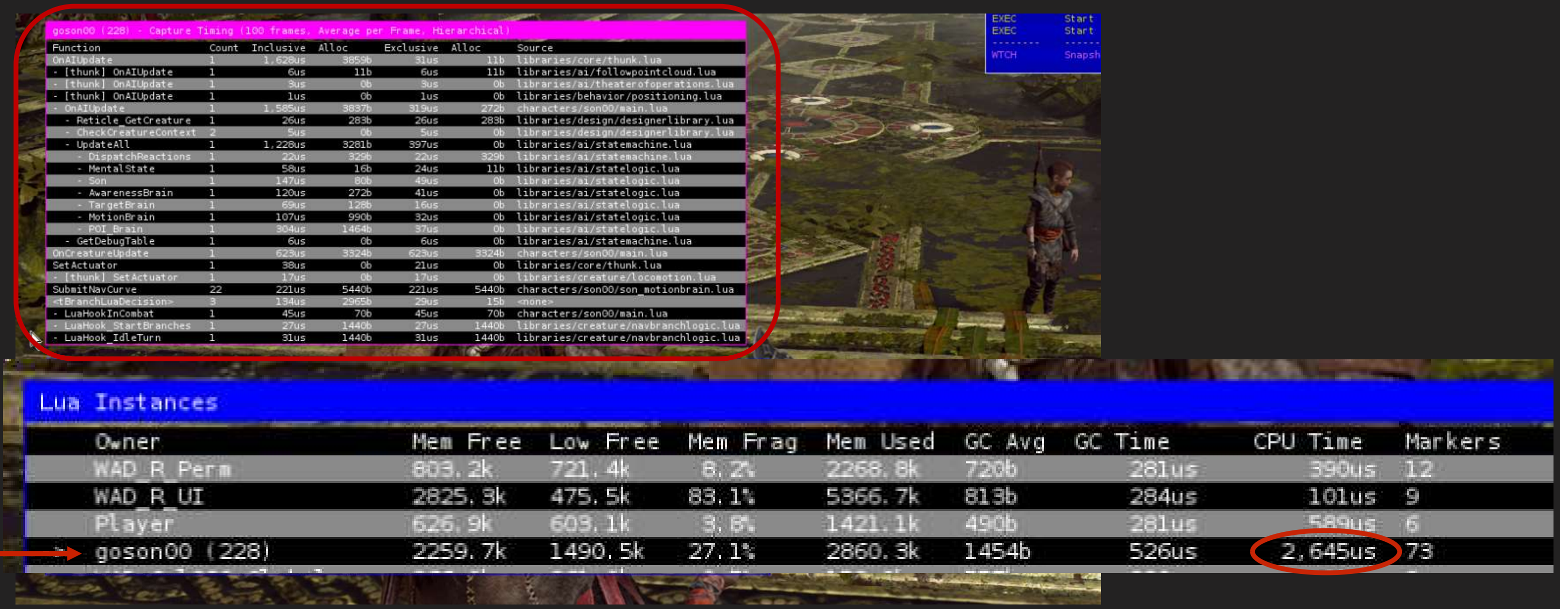
Lua







On top of that, We were using Lua for both (NEXT) AI scripting and (NEXT) level scripting
And Lua as a level scripting solution was being replaced by (NEXT) a Visual Scripting solution for very similar reasons and more, so moving to (NEXT) behavior trees us a chance to reduce the usage of LUA even more (Call to Sam's talk)



```
EXEC      Start
EXEC      Start
--------  -----
WTCH      Snapsh
```

```
goson00 (228) - Capture Timing (100 frames, Average per Frame, Hierarchical)
Function                   Count  Inclusive  Alloc    Exclusive  Alloc   Source
OnAIUpdate                 1        1,628us  3859b        31us     11b   libraries/core/thunk.lua
- [thunk] OnAIUpdate       1            6us    11b         6us     11b   libraries/ai/followpointcloud.lua
- [thunk] OnAIUpdate       1            3us     0b         3us      0b   libraries/ai/theaterofoperations.lua
- [thunk] OnAIUpdate       1            1us     0b         1us      0b   libraries/behavior/positioning.lua
- OnAIUpdate               1        1,585us  3837b       319us    272b   characters/son00/main.lua
  - Reticle_GetCreature    1           26us   283b        26us    283b   libraries/design/designerlibrary.lua
  - CheckCreatureContext   2            5us     0b         5us      0b   libraries/design/designerlibrary.lua
  - UpdateAll              1        1,228us  3281b       397us      0b   libraries/ai/statemachine.lua
    - DispatchReactions    1           22us   329b        22us    329b   libraries/ai/statemachine.lua
    - MentalState          1           58us    16b        24us     11b   libraries/ai/statelogic.lua
    - Son                  1          147us    80b        49us      0b   libraries/ai/statelogic.lua
    - AwarenessBrain       1          120us   272b        41us      0b   libraries/ai/statelogic.lua
    - TargetBrain          1           69us   128b        16us      0b   libraries/ai/statelogic.lua
    - MotionBrain          1          107us   990b        32us      0b   libraries/ai/statelogic.lua
    - POI_Brain            1          304us  1464b        37us      0b   libraries/ai/statelogic.lua
  - GetDebugTable          1            6us     0b         6us      0b   libraries/ai/statemachine.lua
OnCreatureUpdate           1          623us  3324b       623us   3324b   characters/son00/main.lua
SetActuator                1           38us     0b        21us      0b   libraries/core/thunk.lua
- [thunk] SetActuator      1           17us     0b        17us      0b   libraries/creature/locomotion.lua
SubmitNavCurve             22         221us  5440b       221us   5440b   characters/son00/son_motionbrain.lua
<tBranchLuaDecision>       3          134us  2965b        29us     15b   <none>
- LuaHookInCombat          1           45us    70b        45us     70b   characters/son00/main.lua
- LuaHook_StartBranches    1           27us  1440b        27us   1440b   libraries/creature/navbranchlogic.lua
- LuaHook_IdleTurn         1           31us  1440b        31us   1440b   libraries/creature/navbranchlogic.lua
```

```
Lua Instances
    Owner               Mem Free   Low Free   Mem Frag   Mem Used   GC Avg   GC Time   CPU Time   Markers
    WAD_R_Perm          803.2k     721.4k     6.2%       2268.8k    720b       281us      390us   12
    WAD_R_UI            2825.3k    475.5k     83.1%      5366.7k    813b       284us      101us   9
    Player              626.9k     603.1k     3.8%       1421.1k    490b       281us      589us   6
 -> goson00 (228)       2259.7k    1490.5k    27.1%      2860.3k    1454b      526us    2,645us   73
```

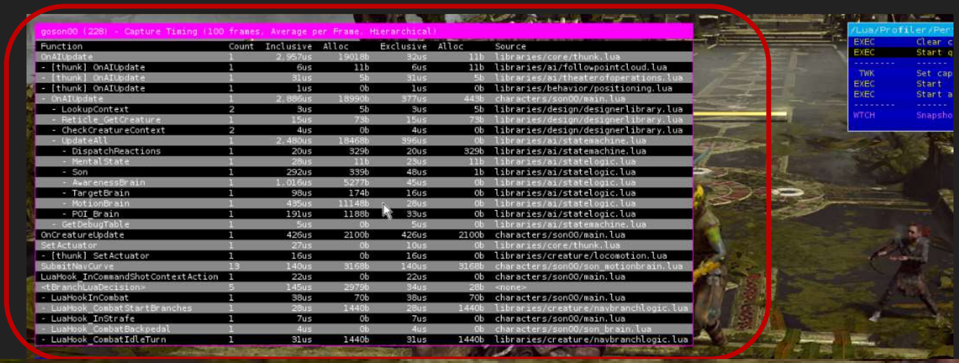
Let me also take a moment to dive a little bit more into issues with LUA

This image here shows lua numbers just for son AI on GOW2018, with the list of things that AI does which was mostly later converted to btree.

- Lua might be considered fast compared to some other options, but it is still a scripting language and we were getting a big hit with the heavy use we are doing for AI behavior. Because we are a studio that relies heavily on tech design, big part of our systems and AI behavior logic was living in lua, most of the time doing very expensive things in update ticks

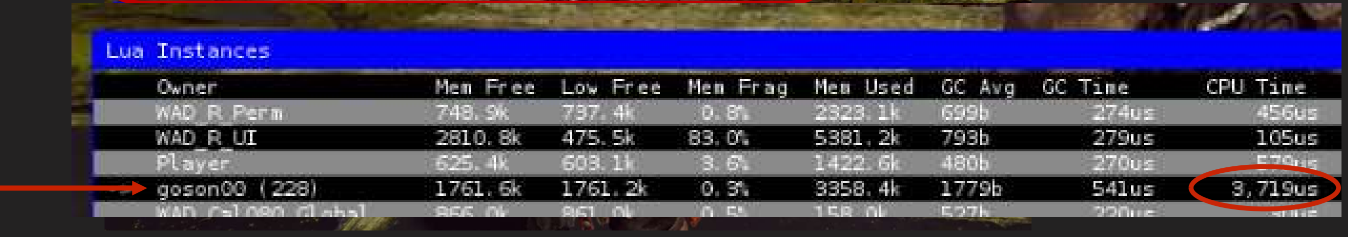
Here you can see out of combat we are looking at avarage around 2.6ms per frame out of combat

Lua → Behavior Tree

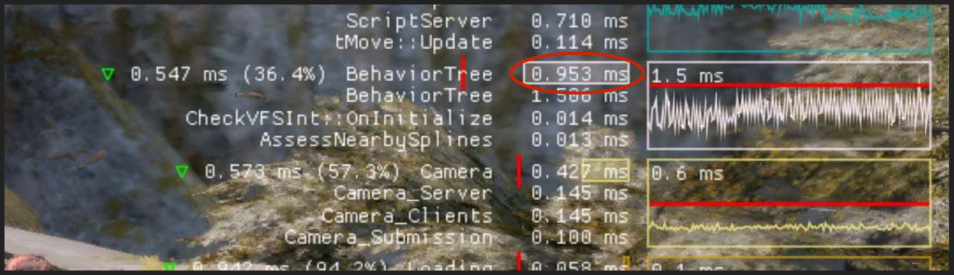


```
goson00 (228) - Capture Timing (100 frames, Average per Frame, Hierarchical)
Function                          Count  Inclusive   Alloc  Exclusive   Alloc  Source
OnAIUpdate                        1        2,957us  19018b       32us     11b  libraries/core/thunk.lua
- [thunk] OnAIUpdate              1            6us     11b        6us     11b  libraries/ai/followpointcloud.lua
- [thunk] OnAIUpdate              1           31us      5b       31us      5b  libraries/ai/theaterofoperations.lua
- [thunk] OnAIUpdate              1            1us      0b        1us      0b  libraries/behavior/positioning.lua
- OnAIUpdate                      1        2,886us  18990b      377us    443b  characters/son00/main.lua
  - LookupContext                 2            3us      5b        3us      5b  libraries/design/designerlibrary.lua
  - Reticle_GetCreature           1           15us     73b       15us     73b  libraries/design/designerlibrary.lua
  - CheckCreatureContext          2            4us      0b        4us      0b  libraries/design/designerlibrary.lua
  - UpdateAll                     1        2,480us  18468b      396us      0b  libraries/ai/statemachine.lua
    - DispatchReactions           1           20us    329b       20us    329b  libraries/ai/statemachine.lua
    - MentalState                 1           28us     11b       23us     11b  libraries/ai/statelogic.lua
    - Son                         1          292us    339b       48us      1b  libraries/ai/statelogic.lua
    - AwarenessBrain              1        1,016us   5277b       45us      0b  libraries/ai/statelogic.lua
    - TargetBrain                 1           98us    174b       16us      0b  libraries/ai/statelogic.lua
    - MotionBrain                 1          435us  11148b       28us      0b  libraries/ai/statelogic.lua
    - POI_Brain                   1          191us   1188b       33us      0b  libraries/ai/statelogic.lua
  - GetDebugTable                 1            5us      0b        5us      0b  libraries/ai/statemachine.lua
OnCreatureUpdate                  1          426us   2100b      426us   2100b  characters/son00/main.lua
SetActuator                       1           27us      0b       10us      0b  libraries/core/thunk.lua
- [thunk] SetActuator             1           16us      0b       16us      0b  libraries/creature/locomotion.lua
SubmitNavCurve                    13         140us   3168b      140us   3168b  characters/son00/son_motionbrain.lua
LuaHook_InCommandShotContextAction 1          22us      0b       22us      0b  characters/son00/main.lua
<tBranchLuaDecision>              5          145us   2979b       34us     28b  <none>
- LuaHookInCombat                 1           38us     70b       38us     70b  characters/son00/main.lua
- LuaHook_CombatStartBranches     1           28us   1440b       28us   1440b  libraries/creature/navbranchlogic.lua
- LuaHook_InStrafe                1            7us      0b        7us      0b  characters/son00/main.lua
- LuaHook_CombatBackpedal         1            4us      0b        4us      0b  characters/son00/son_brain.lua
- LuaHook_CombatIdleTurn          1           31us   1440b       31us   1440b  libraries/creature/navbranchlogic.lua
```

```
/Lua/Profiler/Per
EXEC     Clear c
EXEC     Start q
--------  ------
 TWK     Set cap
EXEC     Start
EXEC     Start a
--------  ------
WTCH     Snapsho
```



```
Lua Instances
     Owner                  Mem Free   Low Free   Mem Frag   Mem Used   GC Avg   GC Time     CPU Time
     WAD_R_Perm             748.9k     737.4k       0.8%     2323.1k    699b       274us        456us
     WAD_R_UI               2810.8k    475.5k      83.0%     5381.2k    793b       279us        105us
     Player                 625.4k     603.1k       3.6%     1422.6k    480b       270us        579us
→    goson00 (228)          1761.6k    1761.2k      0.3%     3358.4k    1779b      541us      3,719us
     WAD_Cal080_Global      866.0k     861.0k       0.5%     158.0k     527b       220us
```



```
                              ScriptServer    0.710 ms
                             tMove::Update    0.114 ms
▽ 0.547 ms (36.4%)            BehaviorTree    0.953 ms    1.5 ms
                              BehaviorTree    1.506 ms
                  CheckVFSInt::OnInitialize    0.014 ms
                       AssessNearbySplines    0.013 ms
      ▽ 0.573 ms (57.3%)            Camera    0.427 ms    0.6 ms
                             Camera_Server    0.145 ms
                            Camera_Clients    0.145 ms
                         Camera_Submission    0.100 ms
     ▽ 0.942 ms (94.2%)            Loading    0.058 ms    0.1 ms
```

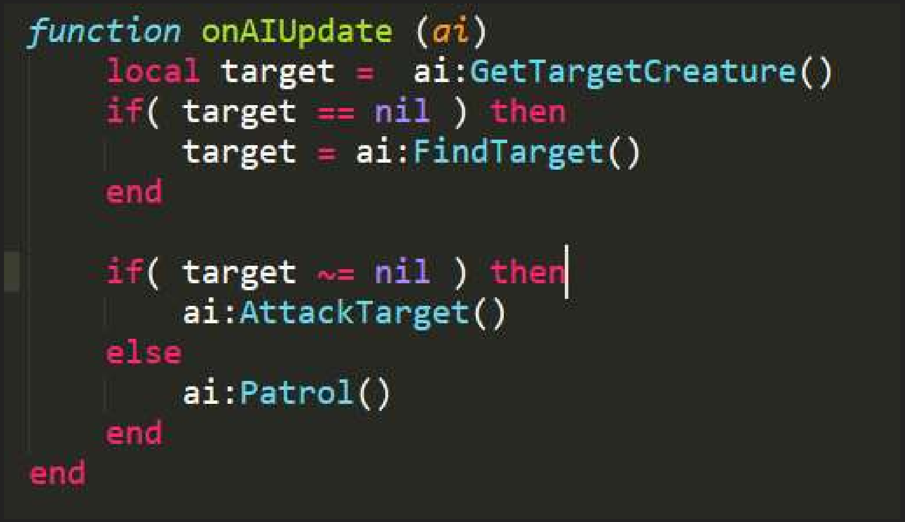
- Performance
- Stability
- Validation & Debugging
- Reading, authoring & maintaining

Which gets worse during combat with 3.7 ms on average. This is obviously before we squeezed every bit of performance we could. But with GOWR and behavior trees, in a similar display we can see that ALL trees run at around ~0.9ms average with peaks up to 1.5. We shipped Ragnarok with a budget of 3ms for btrees with certain setups going over with peak frames here and there.

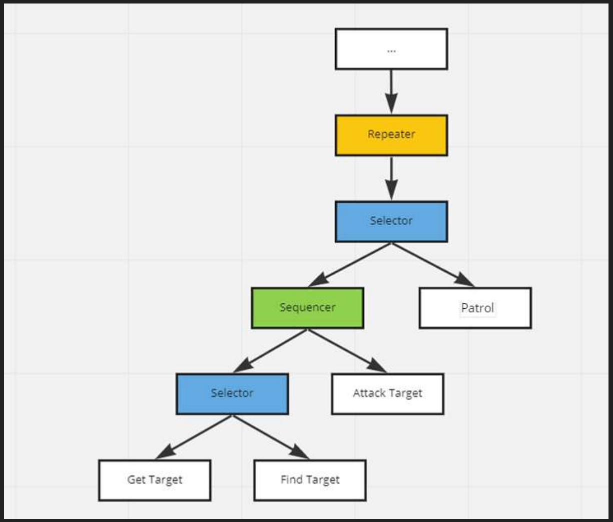
- In addition to performance problem we just talked about, our lua scrips were also not very not stable - we had so many scripts doing a certain thing based on an event, able to make changes at any point in the frame. Tracking bugs were difficult and making sense of sequence of events were confusing
- Lua was difficult to validate in editing time. It was error-prone and we relied on runtime asserts while we lacked reliable debugging tools
- As the functionality in lua grew so did the number of text files, made it difficult to read/author and maintain

Before I dive into how we addressed these problems other than the performance, let me talk a little bit about the process of

Migrating to Behavior Trees

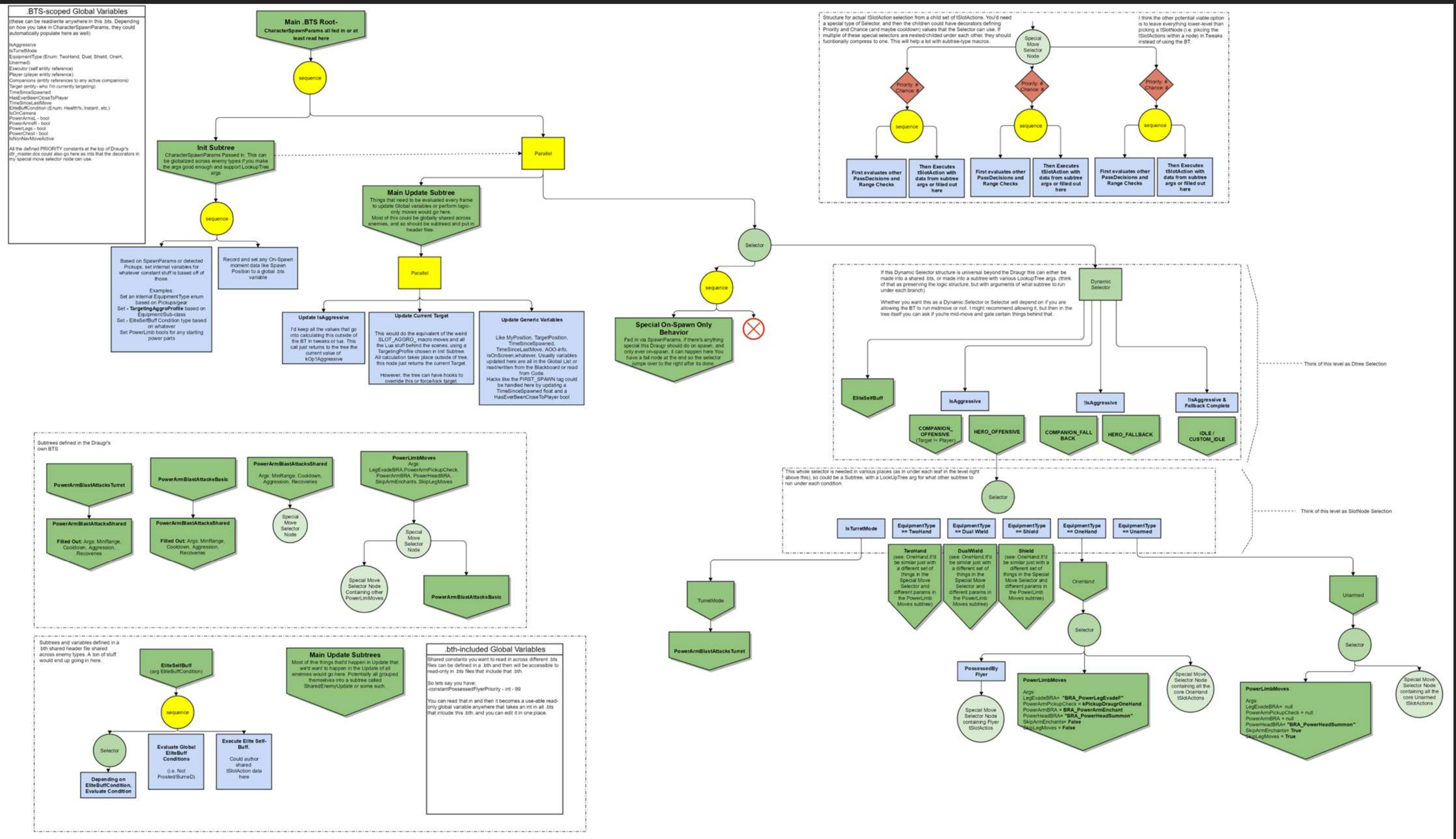


```lua
function onAIUpdate (ai)
    local target =  ai:GetTargetCreature()
    if( target == nil ) then
        target = ai:FindTarget()
    end

    if( target ~= nil ) then
        ai:AttackTarget()
    else
        ai:Patrol()
    end
end
```



... migrating to behavior trees. If you ever had to tell your designers that " we need to throw this thing you are heavily using and have a whole team of trained designers for. And We need to replace it with something completely new that we need to implement from scratch", you know that this is no easy task.

You have seen this very very simplified example of a 1:1 conversation but moving all the content took a lot of effort. And paper planning of it looked a lot more like this :



.BTS-scoped Global Variables

(these can be read/write anywhere in this .bts. Depending on how you take in CharacterSpawnParams, they could automatically populate here as well)

IsAggressive
IsTurretMode
EquipmentType (Enum: TwoHand, Dual, Shield, OneH, Unarmed)
Executor (self entity reference)
Player (player entity reference)
Companions (entity references to any active companions)
Target (entity- who I'm currently targeting)
TimeSinceSpawned
HasEverBeenCloseToPlayer
TimeSinceLastMove
EliteBuffCondition (Enum: Health%, Instant, etc.)
IsOnCamera
PowerArmsL - bool
PowerArmsR - bool
PowerLegs - bool
PowerChest - bool
IsNonNavMoveActive

All the defined PRIORITY constants at the top of Draugr's dtr_master.dcs could also go here as ints that the decorators in my special move selector node can use.

Main .BTS Root-
CharacterSpawnParams all fed in or at least read here

sequence

Init Subtree
CharacterSpawnParams Passed in. This can be globalized across enemy types if you make the args good enough and support LookupTree args

sequence

Based on SpawnParams or detected Pickups, set internal variables for whatever constant stuff is based off of those.

Examples:
Set an internal EquipmentType enum based on Pickups/gear
Set - **TargetingAggroProfile** based on Equipment/Sub-class
Set - EliteSelfBuff Condition type based on whatever
Set PowerLimb bools for any starting power parts

Record and set any On-Spawn moment data like Spawn Position to a global .bts variable

Parallel

Main Update Subtree
Things that need to be evaluated every frame to update Global variables or perform logic-only moves would go here.
Most of this could be globally shared across enemies, and so should be subtreed and put in header files.

Parallel

Update IsAggressive

I'd keep all the values that go into calculating this outside of the BT in tweaks or lua. This call just returns to the tree the current value of kOp1Aggressive

Update Current Target

This would do the equivalent of the weird SLOT_AGGRO_ macro moves and all the Lua stuff behind the scenes, using a TargetingProfile chosen in Init Subtree. All calculation takes place outside of tree, this node just returns the current Target.

However, the tree can have hooks to override this or force/lock target.

Update Generic Variables

Like MyPosition, TargetPosition, TimeSinceSpawned, TimeSinceLastMove, AOO-info, isOnScreen,whatever. Usually variables updated here are all in the Global List or read/written from the Blackboard or read from Code.
Hacks like the FIRST_SPAWN tag could be handled here by updating a TimeSinceSpawned float and a HasEverBeenCloseToPlayer bool

Selector

sequence

Special On-Spawn Only Behavior
Fed in via SpawnParams, if there's anything special this Draugr should do on spawn, and only ever on-spawn, it can happen here.You have a fail node at the end so the selector jumps over to the right after its done.

Structure for actual tSlotAction selection from a child set of tSlotActions. You'd need a special type of Selector, and then the children could have decorators defining Priority and Chance (and maybe cooldown) values that the Selector can use. If multiple of these special selectors are nested/childed under each other, they should fucntionally compress to one. This will help a lot with subtree-type macros.

Special Move Selector Node

I think the other potential viable option is to leave everything lower-level than picking a tSlotNode (i.e. pikcing the tSlotActions within a node) in Tweaks instead of using the BT.

Priority: #
Chance: #

sequence

First evaluates other PassDecisions and Range Checks

Then Executes tSlotAction with data from subtree args or filled out here

Priority: #
Chance: #

sequence

First evaluates other PassDecisions and Range Checks

Then Executes tSlotAction with data from subtree args or filled out here

Priority: #
Chance: #

sequence

First evaluates other PassDecisions and Range Checks

Then Executes tSlotAction with data from subtree args or filled out here

If this Dynamic Selector structure is universal beyond the Draugr this can either be made into a shared .bts, or made into a subtree with various LookupTree args. (think of that as preserving the logic structure, but with arguments of what subtree to run under each branch)

Whether you want this as a Dynamic Selector or Selector will depend on if you are allowing the BT to run midmove or not. I might recommend allowing it, but then in the tree itself you can ask if you're mid-move and gate certain things behind that.

Dynamic Selector

EliteSelfBuff

IsAggressive

COMPANION_OFFENSIVE
(Target != Player)

HERO_OFFENSIVE

!IsAggressive

COMPANION_FALLBACK

HERO_FALLBACK

!IsAggressive & Fallback Complete

IDLE / CUSTOM_IDLE

Think of this level as Dtree Selection

This whole selector is needed in various places (as in under each leaf in the level right above this), so could be a Subtree, with a LookUpTree arg for what other subtree to run under each condition.

Selector

IsTurretMode

EquipmentType == TwoHand

EquipmentType == Dual Wield

EquipmentType == Shield

EquipmentType == OneHand

EquipmentType == Unarmed

Think of this level as SlotNode Selection

TurretMode

PowerArmBlastAttacksTurret

TwoHand
(see: OneHand.It'd be similar just with a different set of things in the Special Move Selector and different params in the PowerLimb Moves subtree)

DualWield
(see: OneHand.It'd be similar just with a different set of things in the Special Move Selector and different params in the PowerLimb Moves subtree)

Shield
(see: OneHand.It'd be similar just with a different set of things in the Special Move Selector and different params in the PowerLimb Moves subtree)

OneHand

Selector

PossessedBy Flyer

Special Move Selector Node containing Flyer tSlotActios

PowerLimbMoves

Args:
LegEvadeBRA= **"BRA_PowerLegEvadeF"**
PowerArmPickupCheck = **kPickupDraugrOneHand**
PowerArmBRA = **BRA_PowerArmEnchant**
PowerHeadBRA= **"BRA_PowerHeadSummon"**
SkipArmEnchants= **False**
SkipLegMoves = **False**

Special Move Selector Node containing all the core OneHand tSlotActions

Unarmed

Selector

PowerLimbMoves

Args:
LegEvadeBRA= null
PowerArmPickupCheck = null
PowerArmBRA = null
PowerHeadBRA= **"BRA_PowerHeadSummon"**
SkipArmEnchants= **True**
SkipLegMoves = **True**

Special Move Selector Node containing all the core Unarmed tSlotActions

Subtrees defined in the Draugr's own BTS

PowerArmBlastAttacksTurret

PowerArmBlastAttacksShared

Filled Out: Args: MinRange, Cooldown, Aggression, Recoveries

PowerArmBlastAttacksBasic

PowerArmBlastAttacksShared

Filled Out: Args: MinRange, Cooldown, Aggression, Recoveries

PowerArmBlastAttacksShared

Args: MinRange, Cooldown, Aggression, Recoveries

Special Move Selector Node

PowerLimbMoves
Args:
LegEvadeBRA,PowerArmPickupCheck, PowerArmBRA, PowerHeadBRA, SkipArmEnchants, SkipLegMoves

Special Move Selector Node

Special Move Selector Node Containing other PowerLimMoves

PowerArmBlastAttacksBasic

Subtrees and variables defined in a .bth shared header file shared across enemy types. A ton of stuff would end up going in here.

EliteSelfBuff
(arg EliteBuffCondition)

sequence

Selector

Depending on EliteBuffCondition, Evaluate Condition

Evaluate Global EliteBuff Conditions

(i.e. Not Frosted/BurneD)

Execute Elite Self-Buff.

Could author shared tSlotAction data here

Main Update Subtrees
Most of thie things that'd happen in Update that we'd want to happen in the Update of all enemies would go here. Potentially all grouped themselves into a subtree called SharedEnemyUpdate or some such.

.bth-included Global Variables

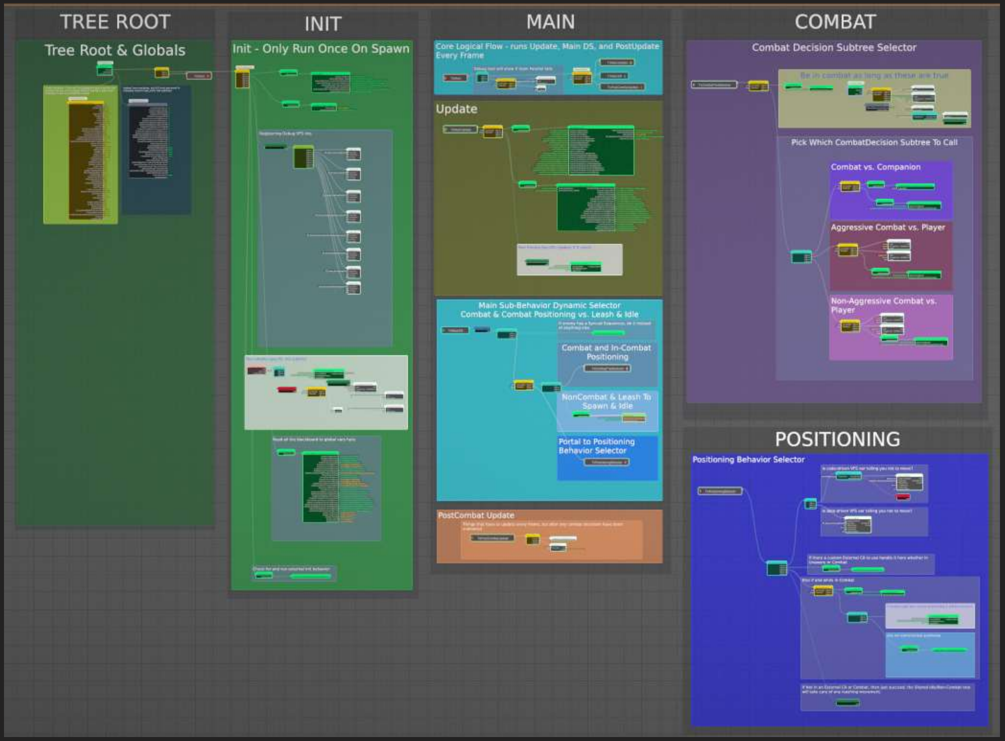
Shared constants you want to read in across different .bts files can be defined in a .bth and then will be accessible to read-only in .bts files that include that .bth.

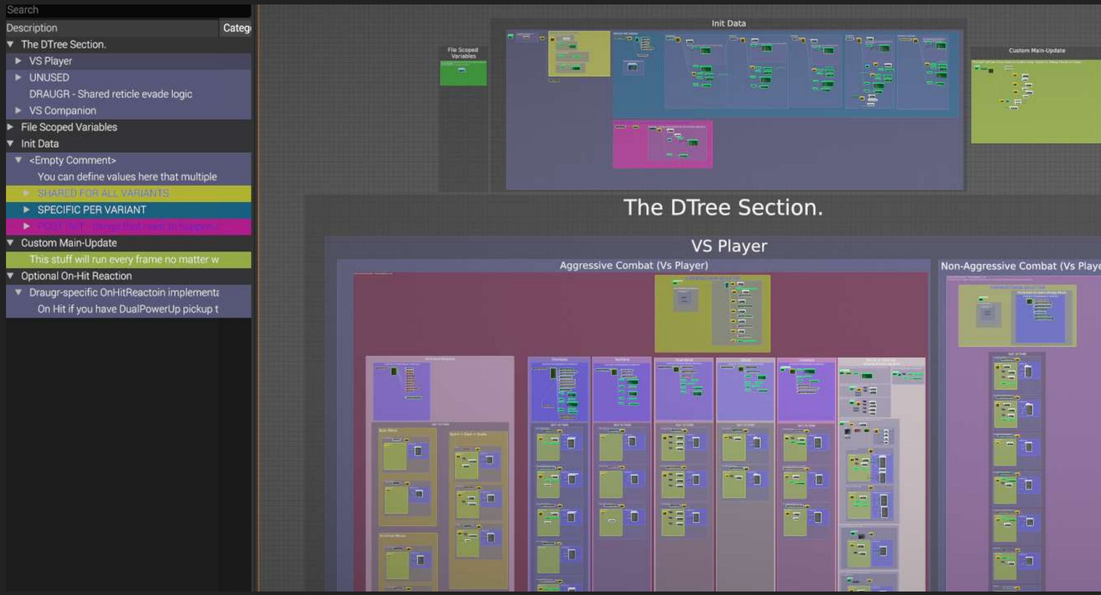
So lets say you have:
-constantPossessedFlyerPriority - int - 99

You can read that in and then it becomes a use-able read-only global variable anywhere that takes an int in all .bts that inlcude this .bth, and you can edit it in one place.

Luckily our now lead combat designer Robert Meyer had a lot of experience with behavior trees so he drove this process. We started from enemies. Lua for enemies had a shared script that was handling all the initialization and updating through a state machine

Migrating to Behavior Trees


TREE ROOT
Tree Root & Globals
INIT
Init - Only Run Once On Spawn
MAIN
Core Logical Flow - runs Update, Main DS, and PostUpdate Every Frame
Update
Main Sub-Behavior Dynamic Selector
Combat & Combat Positioning vs. Leash & Idle
Combat and In-Combat Positioning
NonCombat & Leash To Spawn & Idle
Portal to Positioning Behavior Selector
PostCombat Update
COMBAT
Combat Decision Subtree Selector
Be in combat as long as these are true
Pick Which CombatDecision Subtree To Call
Combat vs. Companion
Aggressive Combat vs. Player
Non-Aggressive Combat vs. Player
POSITIONING
Positioning Behavior Selector

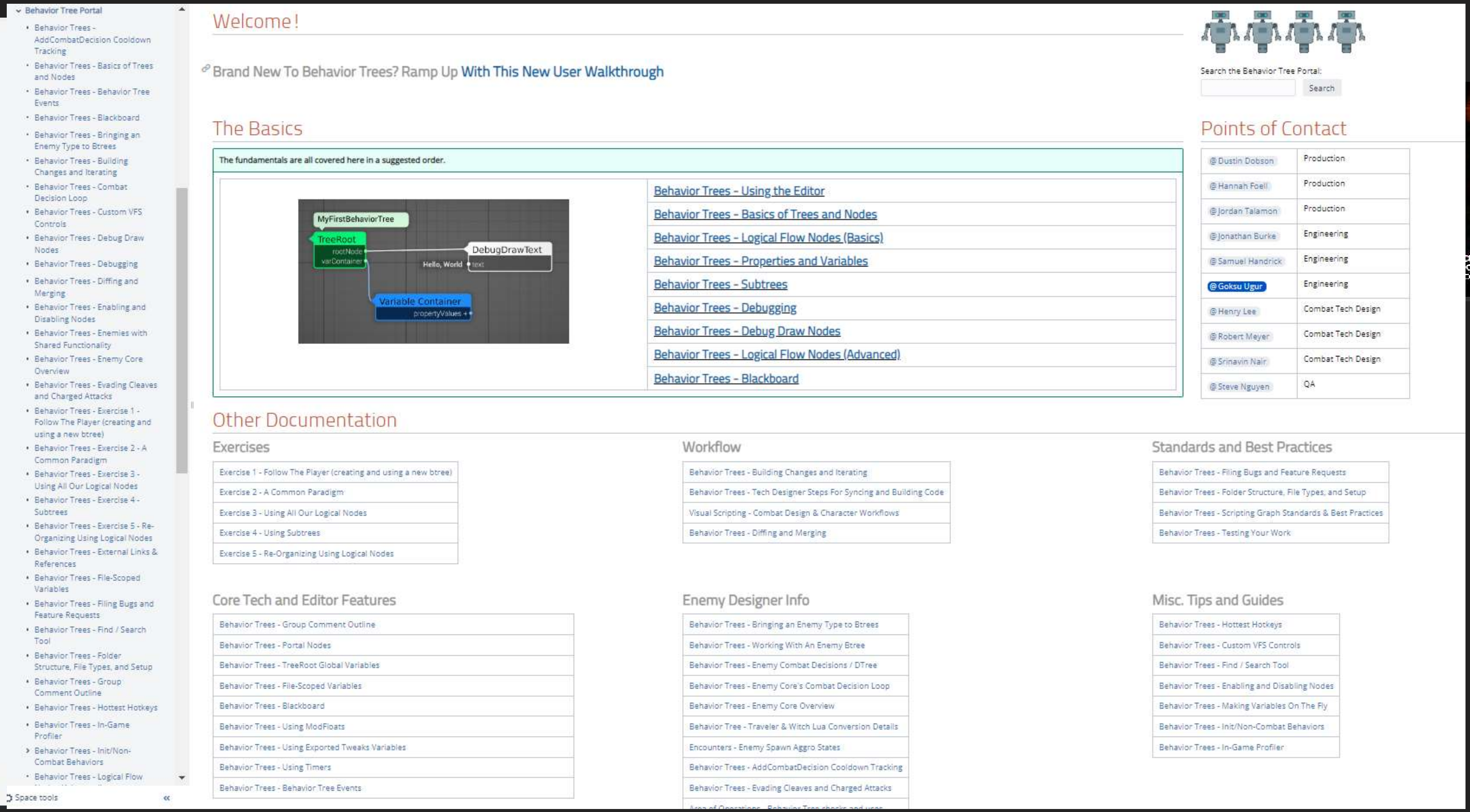


Search

Description | Categ

- ▼ The DTree Section.
  - ▶ VS Player
  - ▶ UNUSED
    - DRAUGR - Shared reticle evade logic
  - ▶ VS Companion
- ▶ File Scoped Variables
- ▼ Init Data
  - ▼ <Empty Comment>
    - You can define values here that multiple
  - ▶ SHARED FOR ALL VARIANTS
  - ▶ SPECIFIC PER VARIANT
  - ▶ POST INIT - things that need to happen
- ▼ Custom Main-Update
  - This stuff will run every frame no matter w
- ▼ Optional On-Hit Reaction
  - ▼ Draugr-specific OnHitReactoin implementa
    - On Hit if you have DualPowerUp pickup t

Lua for enemies had a shared script that was handling all the initialization and updating through a state machine and each enemy type had their own lua file for creature specific setup. Rob spent about 2 months digging into every detail in these files using Draugr as our test case and mapping out a plan. In about 6 months, in parallel with the run time behavior tree system being implemented and the editor and tools, we had our first version of a similar concept with an enemy core behavior tree and draugr subtrees. From that point on we started moving designers over, along with their enemy types and tackling any new issue as it arises.
While there were many upsides to using behavior trees as I will dive more in the rest of the talk, we did have to pay for the cost of a learning curve getting all our designers familiar and efficient in behavior tree. So a lot of effort went into making sure the subtree hookups for each enemy type to be as straightforward as possible. But at the end, behavior trees proved out to be a lot more stable and less error prone when making additions to shared core functionality compared to state machines in lua with its hierarchical nature.
And while of course there were some things that could have been handled better, a common post mortem feedback indicates that the process of introduction&transition to behaviortrees was mostly a successful one. And I think the most important reason for that was that we focused on Build AI systems & tools WITH designers instead of just FOR them



- Behavior Tree Portal
  - Behavior Trees - AddCombatDecision Cooldown Tracking
  - Behavior Trees - Basics of Trees and Nodes
  - Behavior Trees - Behavior Tree Events
  - Behavior Trees - Blackboard
  - Behavior Trees - Bringing an Enemy Type to Btrees
  - Behavior Trees - Building Changes and Iterating
  - Behavior Trees - Combat Decision Loop
  - Behavior Trees - Custom VFS Controls
  - Behavior Trees - Debug Draw Nodes
  - Behavior Trees - Debugging
  - Behavior Trees - Diffing and Merging
  - Behavior Trees - Enabling and Disabling Nodes
  - Behavior Trees - Enemies with Shared Functionality
  - Behavior Trees - Enemy Core Overview
  - Behavior Trees - Evading Cleaves and Charged Attacks
  - Behavior Trees - Exercise 1 - Follow The Player (creating and using a new btree)
  - Behavior Trees - Exercise 2 - A Common Paradigm
  - Behavior Trees - Exercise 3 - Using All Our Logical Nodes
  - Behavior Trees - Exercise 4 - Subtrees
  - Behavior Trees - Exercise 5 - Re-Organizing Using Logical Nodes
  - Behavior Trees - External Links & References
  - Behavior Trees - File-Scoped Variables
  - Behavior Trees - Filing Bugs and Feature Requests
  - Behavior Trees - Find / Search Tool
  - Behavior Trees - Folder Structure, File Types, and Setup
  - Behavior Trees - Group Comment Outline
  - Behavior Trees - Hottest Hotkeys
  - Behavior Trees - In-Game Profiler
  - Behavior Trees - Init/Non-Combat Behaviors
  - Behavior Trees - Logical Flow

Space tools

Welcome!

Brand New To Behavior Trees? Ramp Up With This New User Walkthrough

Search the Behavior Tree Portal:

Search

The Basics

The fundamentals are all covered here in a suggested order.

- Behavior Trees - Using the Editor
- Behavior Trees - Basics of Trees and Nodes
- Behavior Trees - Logical Flow Nodes (Basics)
- Behavior Trees - Properties and Variables
- Behavior Trees - Subtrees
- Behavior Trees - Debugging
- Behavior Trees - Debug Draw Nodes
- Behavior Trees - Logical Flow Nodes (Advanced)
- Behavior Trees - Blackboard

Points of Contact

@Dustin Dobson	Production
@Hannah Foell	Production
@Jordan Talamon	Production
@Jonathan Burke	Engineering
@Samuel Handrick	Engineering
@Goksu Ugur	Engineering
@Henry Lee	Combat Tech Design
@Robert Meyer	Combat Tech Design
@Srinavin Nair	Combat Tech Design
@Steve Nguyen	QA

Other Documentation

Exercises

- Exercise 1 - Follow The Player (creating and using a new btree)
- Exercise 2 - A Common Paradigm
- Exercise 3 - Using All Our Logical Nodes
- Exercise 4 - Using Subtrees
- Exercise 5 - Re-Organizing Using Logical Nodes

Workflow

- Behavior Trees - Building Changes and Iterating
- Behavior Trees - Tech Designer Steps For Syncing and Building Code
- Visual Scripting - Combat Design & Character Workflows
- Behavior Trees - Diffing and Merging

Standards and Best Practices

- Behavior Trees - Filing Bugs and Feature Requests
- Behavior Trees - Folder Structure, File Types, and Setup
- Behavior Trees - Scripting Graph Standards & Best Practices
- Behavior Trees - Testing Your Work

Core Tech and Editor Features

- Behavior Trees - Group Comment Outline
- Behavior Trees - Portal Nodes
- Behavior Trees - TreeRoot Global Variables
- Behavior Trees - File-Scoped Variables
- Behavior Trees - Blackboard
- Behavior Trees - Using ModFloats
- Behavior Trees - Using Exported Tweaks Variables
- Behavior Trees - Using Timers
- Behavior Trees - Behavior Tree Events

Enemy Designer Info

- Behavior Trees - Bringing an Enemy Type to Btrees
- Behavior Trees - Working With An Enemy Btree
- Behavior Trees - Enemy Combat Decisions / DTree
- Behavior Trees - Enemy Core's Combat Decision Loop
- Behavior Trees - Enemy Core Overview
- Behavior Tree - Traveler & Witch Lua Conversion Details
- Encounters - Enemy Spawn Aggro States
- Behavior Trees - AddCombatDecision Cooldown Tracking
- Behavior Trees - Evading Cleaves and Charged Attacks
- Area of Operations - Behavior Tree checks and uses

Misc. Tips and Guides

- Behavior Trees - Hottest Hotkeys
- Behavior Trees - Custom VFS Controls
- Behavior Trees - Find / Search Tool
- Behavior Trees - Enabling and Disabling Nodes
- Behavior Trees - Making Variables On The Fly
- Behavior Trees - Init/Non-Combat Behaviors
- Behavior Trees - In-Game Profiler

Here are some things that I think were key to this transition:

Do not start from scratch – Lucky for us Geurilla was kind enough to give us the Decima framework to start building our Btree editor on top of. So at the very least on the editor side, we were able to get a head start. it saved us A LOT of time, and we were able to build our edotor tool that we branded as Forge

Create foundational nodes as quickly as possible (not only btree foundational nodes like sequencers, selectors but also data getters, core functionality setters for positioning, animation and combat etc anything that will be used) so that design team has enough to continue building with

Devoted design group to be constantly checking new features, actively creating core btrees, giving usability feedback. The whole design team did not need to be involved, in fact it was better to move faster with a smaller focused group and an environment of quick turnaround and iteration

To be able to do that we had to allow for usages of both systems until both programming and design teams are ready. With an easy run time switch option. This way not only we can easily parity test things, we also allowed our design department continue working on prototyping uninterrupted in familiar systems while things are in progress. But that also meant they would have to be rolled over at some point

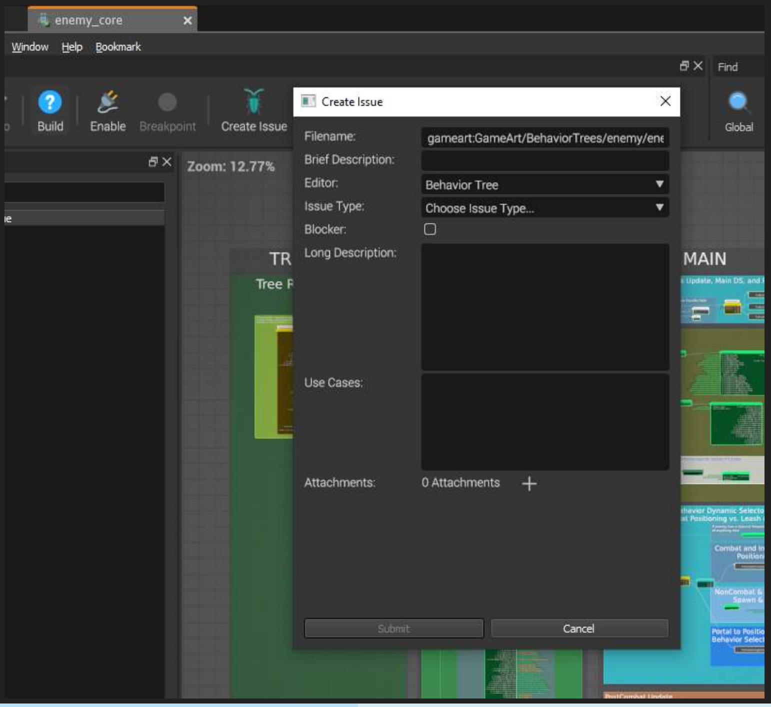
So we built documentation as we built the system :

We had devoted wiki pages that had the tech proposals, roadmaps and detailed information for programmers. But also a behavior tree page devoted to new users covering many aspects of transition from basics, to tutorials, best practices to editor features, each with a lot of visual aid. Because getting newcomers up to speed as easy as possible was very important for our success and we planned for it and we held each other accountable for updating it as things changed

Migrating to Behavior Trees

- Get a head start if you can
- Create foundational nodes first
- Devoted design support
- Allow usages of both systems
- Build documentation as you build the system
- In tool issue reporting button
- Production support
- QA support

DECIMA



Create Issue

Filename: gameart:GameArt/BehaviorTrees/enemy/ene

Brief Description:

Editor: Behavior Tree ▾

Issue Type: Choose Issue Type... ▾

Blocker: □

Long Description:

Use Cases:

Attachments: 0 Attachments +

Submit | Cancel

In tool issue reporting button (simplest but biggest win!) that automatically created a jira based on selected options–

Eliminates things slipping through cracks, minimizes effort needed for issue reporting and thus made it more likely that we would hear about problems .

The button is good to know about all the issues users are facing, but it also means you have so many jiras now. The in tool button is only effective if you have good production support. You need to constantly triage, prioritize, assign and plan for fix. Even without the button a good production support and regular syncs was crucial for us early on for tracking and visibility across teams

QA support on tools – on top of the usability feedback we get from designers QA was crucial to keep the floor stable and catch tool bugs before the builds are released

Speaking of Tools ...

- Behavior Tree Editor
- Debugger
- Diff/Merge Tool
- Performance Tool
- Runtime Debug Options

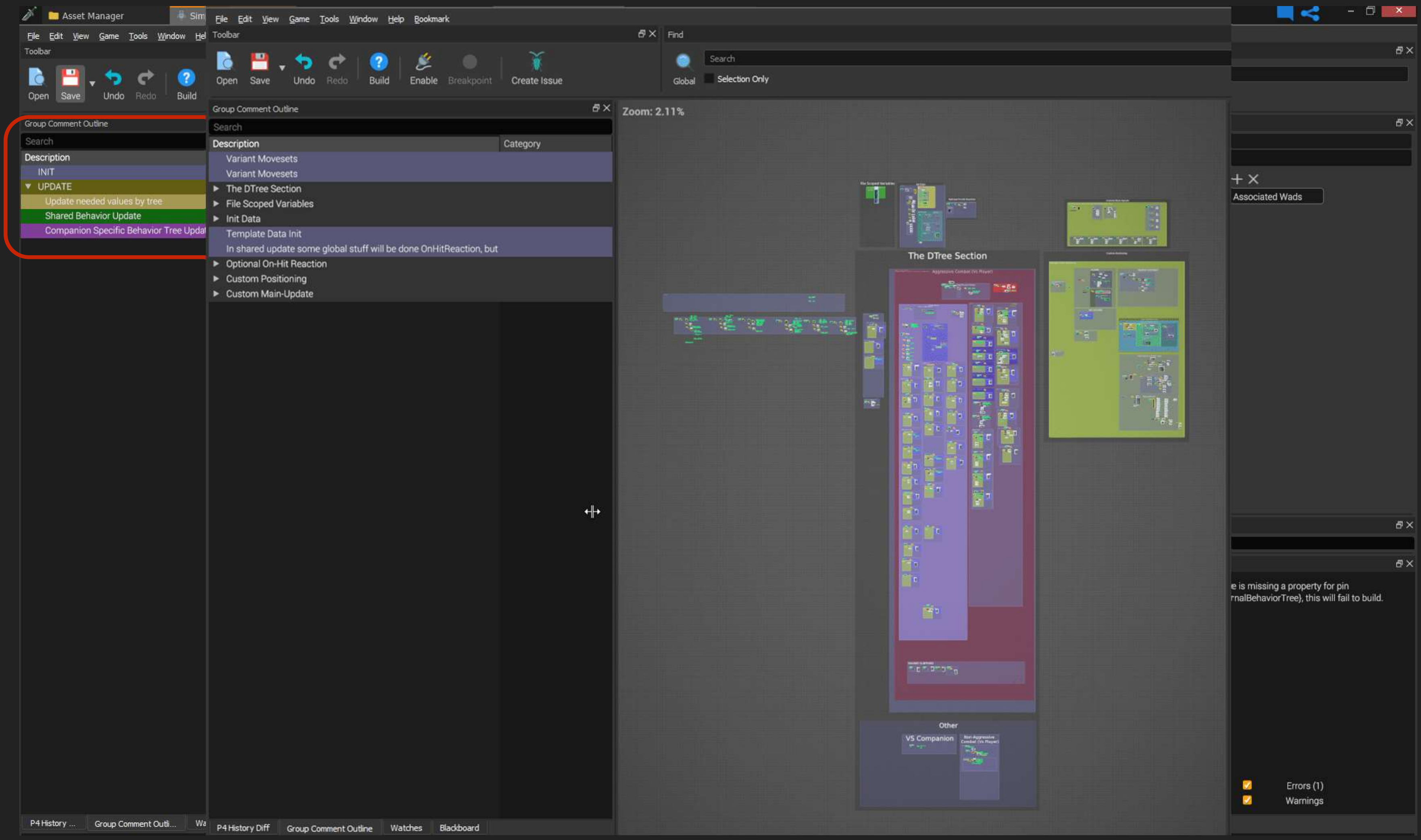
Problems to solve :

- Performance
- Stability
- Validation & Debugging
- Reading, authoring & maintaining

Santa Monica Studio

Having a good suit of tools were crucial for us to get our completely new Behavior Tree system up and running successfully. They did not all come online early on of course, but at least an earlier version of them was available when needed and we continued iterating through the project. Building tools and training users as we were building these systems and moving content over in parallel was challenging. But they played a big role in keeping partner teams convinced and content. But more importantly they were crucial to make sure we were solving the problems we were set out to do

Behavior Tree Editor - Forge



Group Comment Outline

Search

Description
INIT
▼ UPDATE
Update needed values by tree
Shared Behavior Update
Companion Specific Behavior Tree Updat

Group Comment Outline

Search

Description	Category
Variant Movesets	
Variant Movesets	
▶ The DTree Section	
▶ File Scoped Variables	
▶ Init Data	
Template Data Init	
In shared update some global stuff will be done OnHitReaction, but	
▶ Optional On-Hit Reaction	
▶ Custom Positioning	
▶ Custom Main-Update	

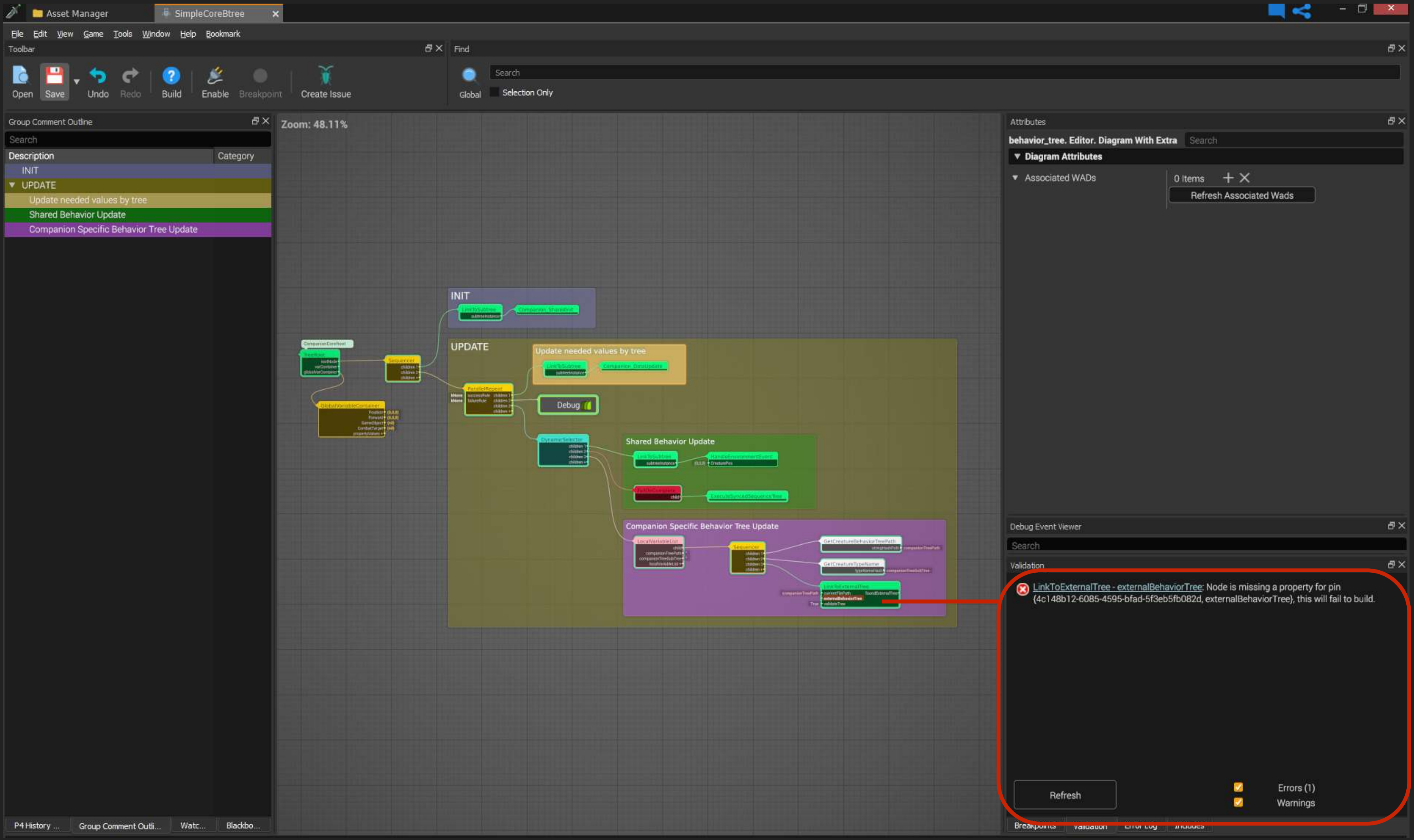
e is missing a property for pin
rnalBehaviorTree), this will fail to build.

☑ Errors (1)
☑ Warnings

Santa Monica Studio

This is what it looked like in a nutshell. : As a side effect of Frankesteining an existing editor tool, it does not go from top to bottom like a regular tree, but it goes left to right. Huge thanks to Josh Phelan, Sam Sternklar and Jeff Miller for their hard work on many aspects of this tool
Fun fact : the very first version of the btree was fully text based and had no editor. We had in game screen draw with boxes for the visualization purposes. Shoutout to our gameplay director Jon Burk for doing a lot of heavy lifting getting this up and running early on . And Aitor Santamaria Ibirika and Sam Handrick for their continuous hard work on it
Goal : Have it be easy to organize/navigate/read/author. So our focus was to Optimize designer time with the tool
The regular expected stuff: , search for nodes, make connections….Coloring of nodes to give users a quick idea of the tree at a glance
- And there were also many more quality of life improvements and many more shortcuts added here with feedback from all the users
But I would like to go over some small improvements that build up to bigger differences for us and things that addressed our issues with lua text files
- Grouping comments – they were used similar to code comments but provided more– helps organizing and easy hierarchy look (panel on the left) also navigates to block when clicked. It might seem like a simple little thing but when you think about how bug and complicated behavior trees might get, they came quiet handy

Behavior Tree Editor - Forge



Description	Category
INIT	
▼ UPDATE	
Update needed values by tree	
Shared Behavior Update	
Companion Specific Behavior Tree Update	

behavior_tree. Editor. Diagram With Extra

▼ Diagram Attributes

▼ Associated WADs — 0 Items

Refresh Associated Wads

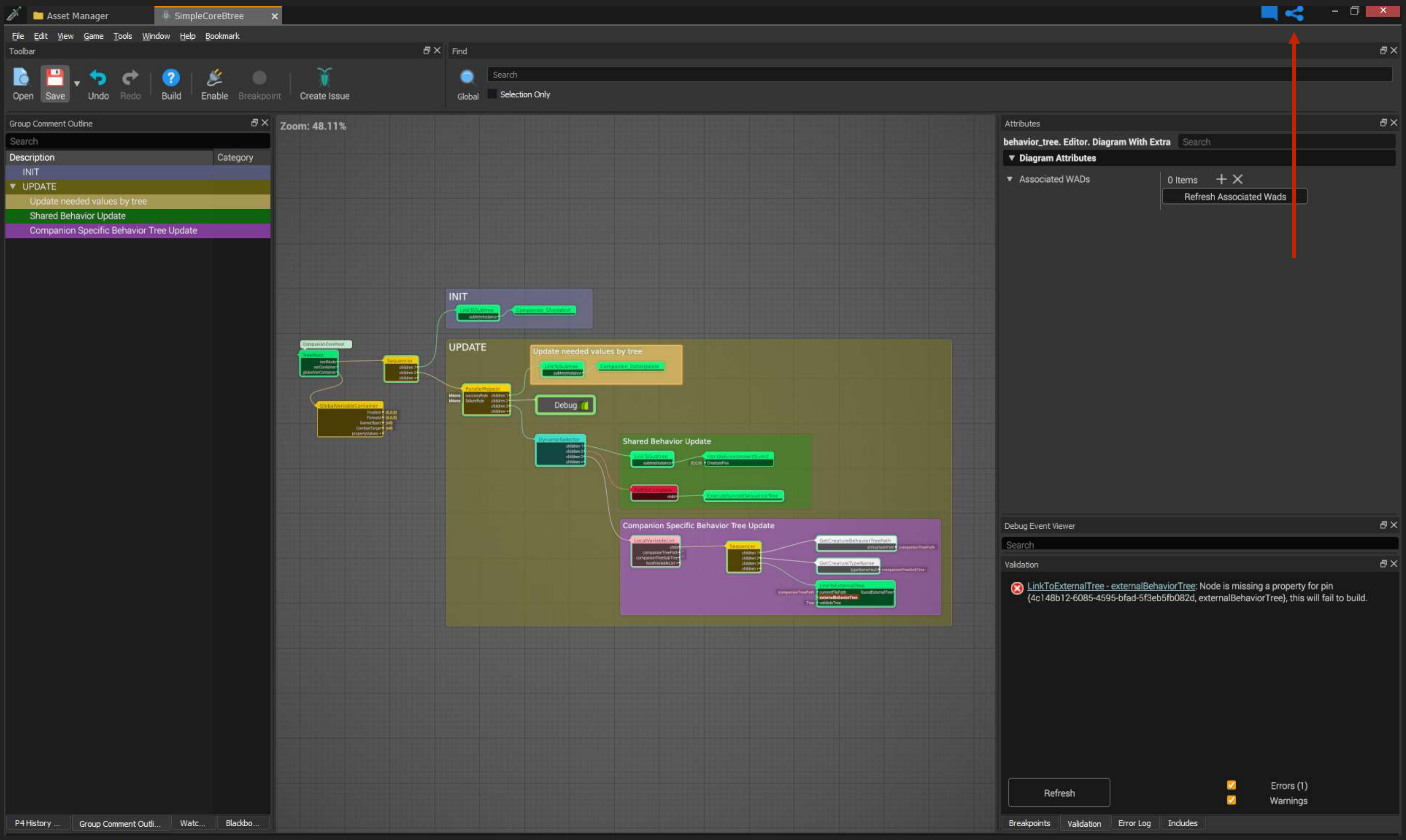
LinkToExternalTree - externalBehaviorTree: Node is missing a property for pin {4c148b12-6085-4595-bfad-5f3eb5fb082d, externalBehaviorTree}, this will fail to build.

☑ Errors (1)
☑ Warnings

Santa Monica Studio

- We also supported in tool data validation as you can see here as an improvement upon the mostly runtime error catching of lua

Behavior Tree Editor - Forge

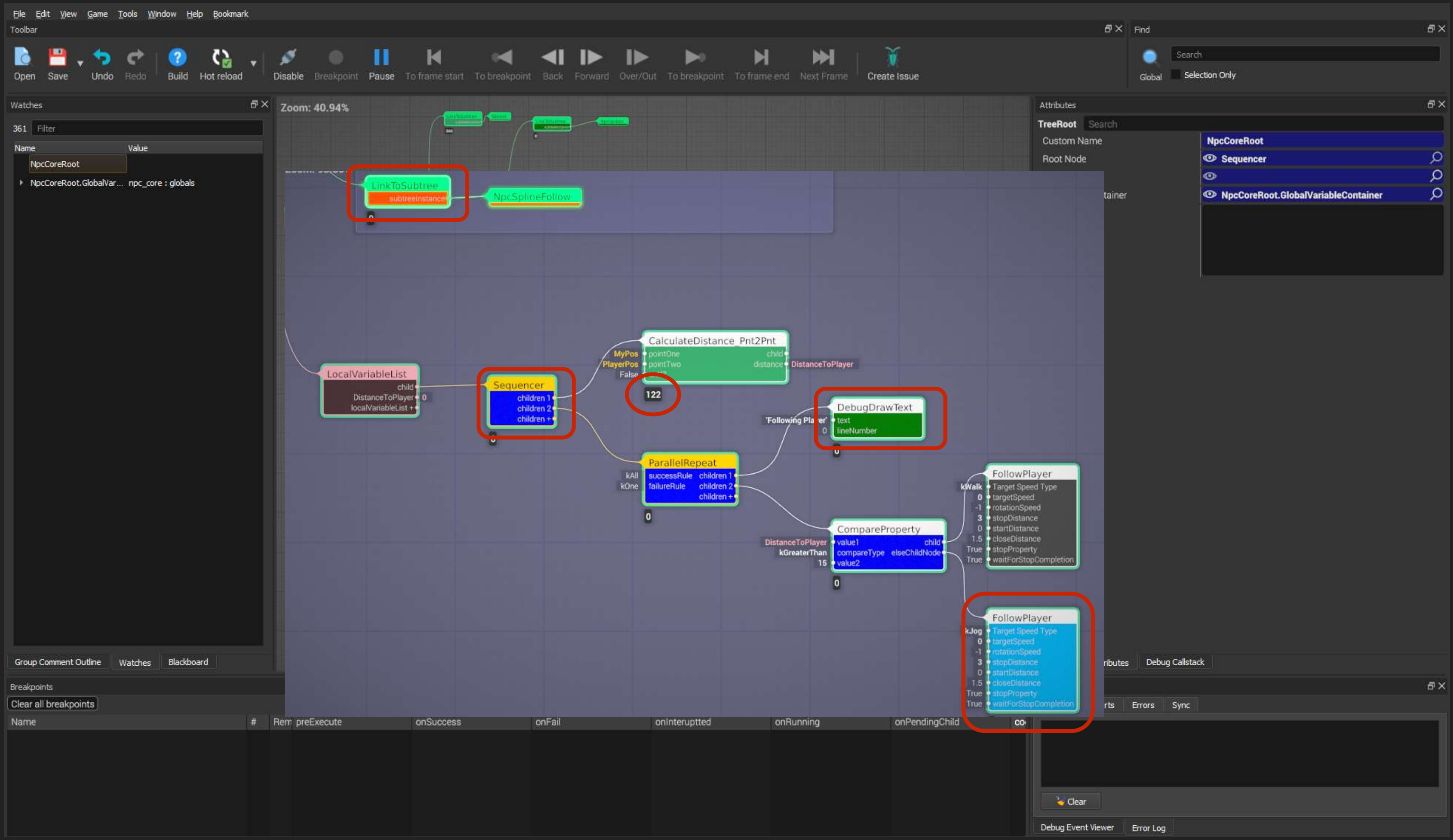


LinkToExternalTree - externalBehaviorTree: Node is missing a property for pin {4c148b12-6085-4595-bfad-5f3eb5fb082d, externalBehaviorTree}, this will fail to build.

Santa Monica Studio

- We also had the ability of sharing a link to an existing view of a file. This button generated a text that had the info on the file path, the focused node and position in the canvas. This way anyone can click or paste would automatically be taken to that point in the tool
 – super easy/fast to pass things around when talking about things, troubleshooting (especially in a WFH setup)

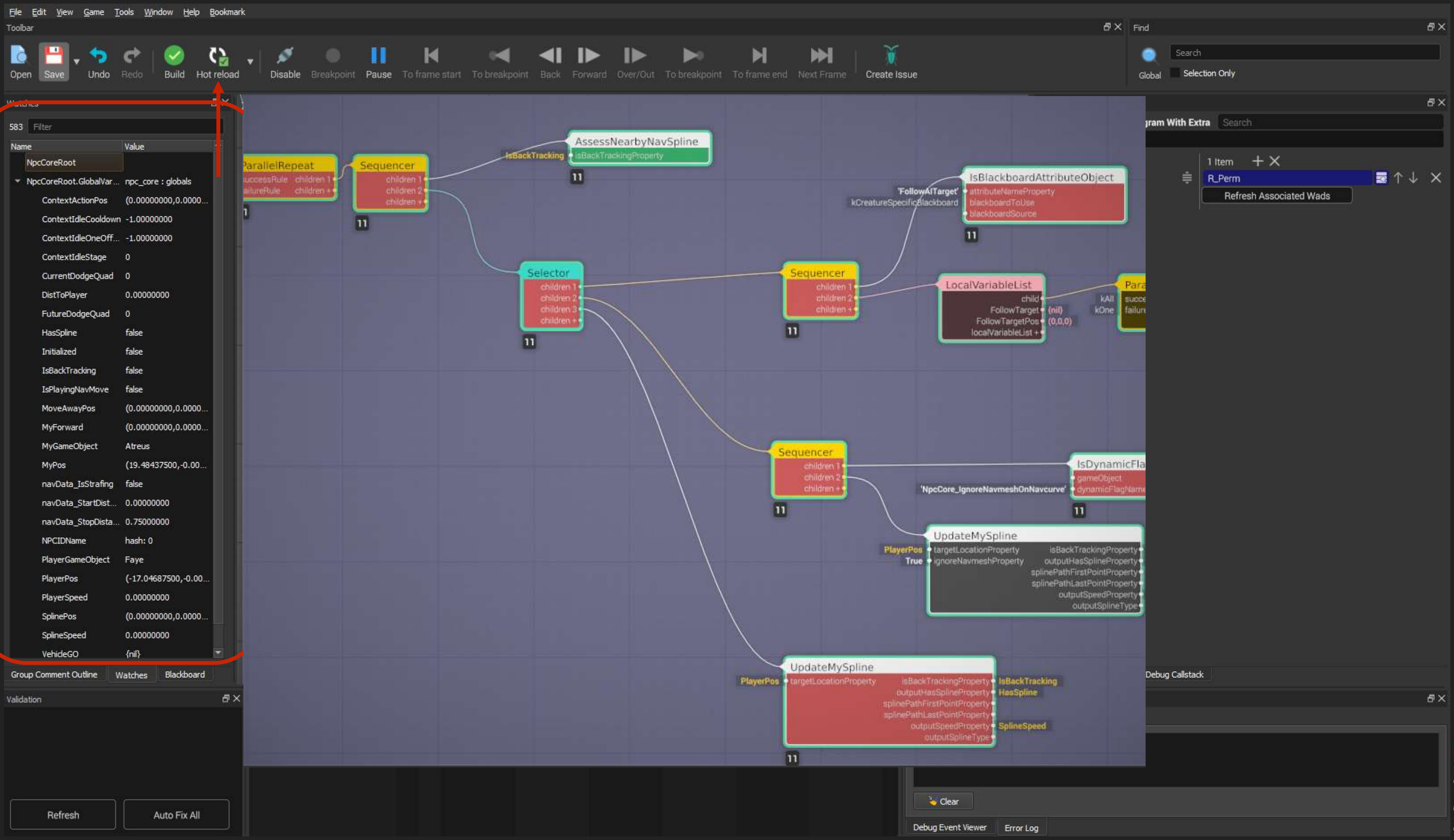
Behavior Tree Debugger



Goal : Have reliable debugging with breakpoint support while accommodating multiple options across different use cases. This was one of the bigger wins we had over lua

- The coloring is an easy way to tell what states nodes are in one look. So if we were to look at this section of btree closer. Red indicates a node that failed this frame, green succeeded, light blue actively running (meaning the task it is doing spans across multiple frames)and dark blue pending child. The number at the bottom represents the amount of frames since the last time this node was updated, accommodated by light green means that the last time this node ran was112 frames ago and it succeeded. This kind of information was crucial tracking bugs

Behavior Tree Debugger



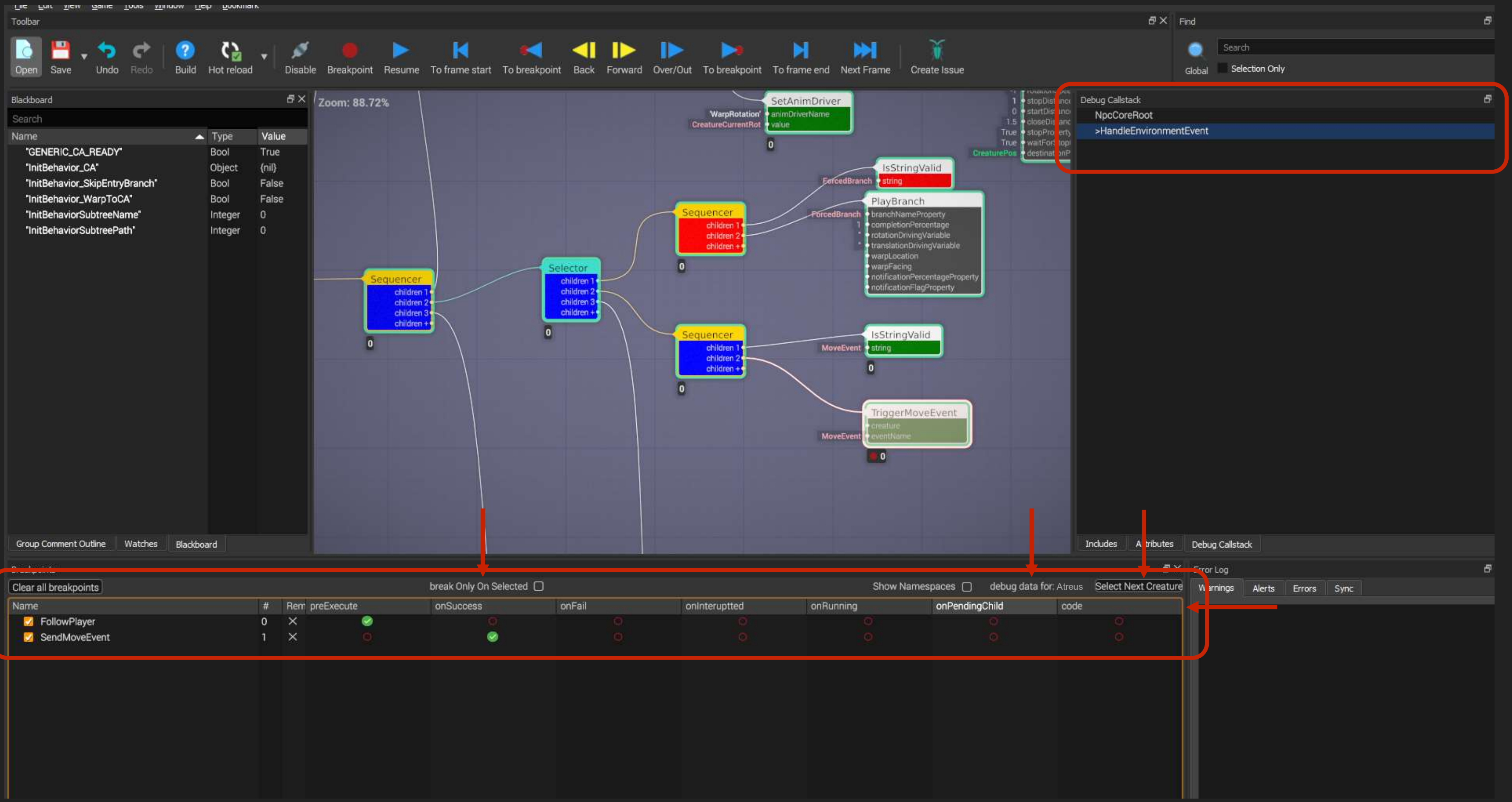
583 Filter

Name	Value
NpcCoreRoot	
NpcCoreRoot.GlobalVar...	npc_core : globals
ContextActionPos	(0.00000000,0.0000...
ContextIdleCooldown	-1.00000000
ContextIdleOneOff...	-1.00000000
ContextIdleStage	0
CurrentDodgeQuad	0
DistToPlayer	0.00000000
FutureDodgeQuad	0
HasSpline	false
Initialized	false
IsBackTracking	false
IsPlayingNavMove	false
MoveAwayPos	(0.00000000,0.0000...
MyForward	(0.00000000,0.0000...
MyGameObject	Atreus
MyPos	(19.48437500,-0.00...
navData_IsStrafing	false
navData_StartDist...	0.00000000
navData_StopDista...	0.75000000
NPCIDName	hash: 0
PlayerGameObject	Faye
PlayerPos	(-17.04687500,-0.00...
PlayerSpeed	0.00000000
SplinePos	(0.00000000,0.0000...
SplineSpeed	0.00000000
VehicleGO	{nil}

So if for example we get a bug with companion left behind too far away from player, by just looking at this behavior tree debug at the time we can tell that companion was in the process of getting to player but was interrupted (NEXT) 10 frames ago since this node is pink. And there is a current Environment even the companion is still in the process of resolving. We can also tell that the same frame there was no context action or spline for companion to engage in seeing the last time these ran they failed. Frame numbers help because we can jump into any subtree and cross reference what happened in that specific frame to get more information

- Watch window for btree variables and the blackboard values. Blackboard values were used for more cross system communication or for easy access across different callstacks
- Hot Reload – adds the ability to update the game with the changes in betree files while the game runs without needing to realod the game or respawn AI (massive help on iteration times)

Behavior Tree Debugger



Blackboard

Search

Name	Type	Value
"GENERIC_CA_READY"	Bool	True
"InitBehavior_CA"	Object	{nil}
"InitBehavior_SkipEntryBranch"	Bool	False
"InitBehavior_WarpToCA"	Bool	False
"InitBehaviorSubtreeName"	Integer	0
"InitBehaviorSubtreePath"	Integer	0

Debug Callstack

NpcCoreRoot
>HandleEnvironmentEvent

Clear all breakpoints | break Only On Selected □ | Show Namespaces □ | debug data for: Atreus | Select Next Creature

Name	#	Rem	preExecute	onSuccess	onFail	onInteruptted	onRunning	onPendingChild	code
☑ FollowPlayer	0	×	●	○	○	○	○	○	○
☑ SendMoveEvent	1	×	○	●	○	○	○	○	○

- We also had full breakpoint support with some different options. When they are hit, users can step through, jump to next breakpoint or resume etc

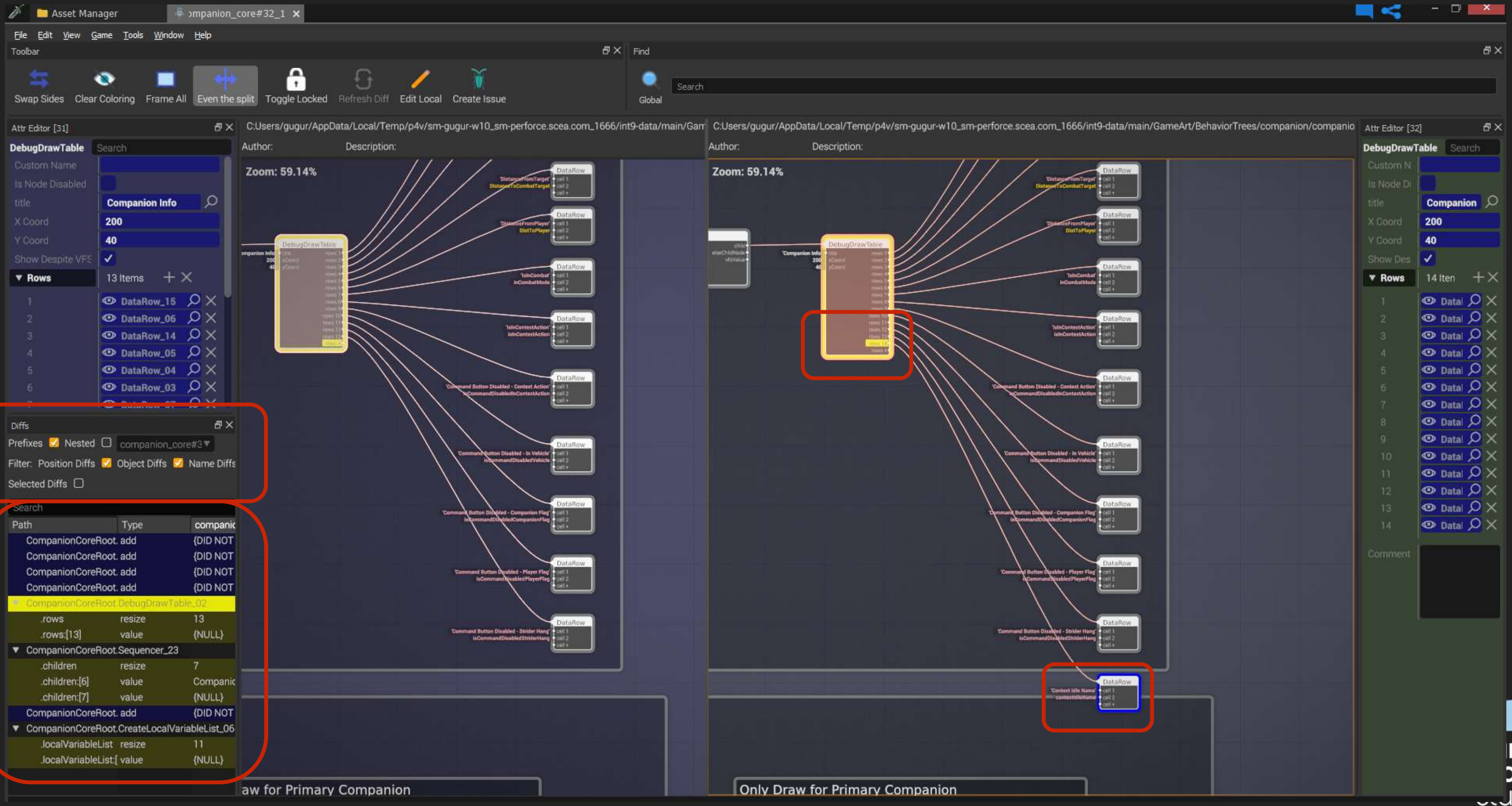
Our runtime behavior tree actually did not hold the record of the callstack, it only kept the currently running tasks for memory and performance gains. But since that means we never crawl from parent to children each frame, but instead go from running task to running task, just using processes nodes in runtime for debugger resulted in incomprehensible jumps from nodes to nodes. So we held the additional info to simulate crawling through the tree each frame for users.

- Similar debug info also helped us with this debug callstack window, to help clear things out when same subtrees being called from different points in the tree

-if you look here (you also see different breakpoint options, which acts as conditional breakpoints for node state. In editor breakpoints actually hit at the end of the frame after btree is processed. and they are not real breakpoints, they just zero time the game, an ability we use to freeze the time and some system while the game is running. This way, It gives the user the ability to scrub through each node, forward and back, watch values state changes, recreate the events of that frame. Currently we do not have support to go backward to a previous frame, but future plans include this ability

- That also meant we also provide "pre execute code breakpoint" for easier programmer debugging- which is an actual forced code breakpoint that happens in the corresponding code for that the node

Diff/Merge Tool



C:Users/gugur/AppData/Local/Temp/p4v/sm-gugur-w10_sm-perforce.scea.com_1666/int9-data/main/Gam

C:Users/gugur/AppData/Local/Temp/p4v/sm-gugur-w10_sm-perforce.scea.com_1666/int9-data/main/GameArt/BehaviorTrees/companion/companio

Attr Editor [31]

DebugDrawTable

Custom Name	
Is Node Disabled	□
title	Companion Info
X Coord	200
Y Coord	40
Show Despite VFS	✓
▼ Rows	13 Items
1	DataRow_15
2	DataRow_06
3	DataRow_14
4	DataRow_05
5	DataRow_04
6	DataRow_03

Diffs

Prefixes ☑ Nested □ companion_core#3

Filter: Position Diffs ☑ Object Diffs ☑ Name Diffs

Selected Diffs □

Path	Type	companic
CompanionCoreRoot.	add	{DID NOT
CompanionCoreRoot.	add	{DID NOT
CompanionCoreRoot.	add	{DID NOT
CompanionCoreRoot.	add	{DID NOT
▼ CompanionCoreRoot.DebugDrawTable_02		
.rows	resize	13
.rows:[13]	value	{NULL}
▼ CompanionCoreRoot.Sequencer_23		
.children	resize	7
.children:[6]	value	Companic
.children:[7]	value	{NULL}
CompanionCoreRoot.	add	{DID NOT
▼ CompanionCoreRoot.CreateLocalVariableList_06		
.localVariableList	resize	11
.localVariableList:[	value	{NULL}

Attr Editor [32]

DebugDrawTable

Custom N	
Is Node Di	□
title	Companion
X Coord	200
Y Coord	40
Show Des	✓
▼ Rows	14 Iten
1	Datal
2	Datal
3	Datal
4	Datal
5	Datal
6	Datal
7	Datal
8	Datal
9	Datal
10	Datal
11	Datal
12	Datal
13	Datal
14	Datal
Comment	

Goal : This was one of the downsides of switching to behavior trees. These files were being saved text files that were optimized for faster processing when loading and reading but it was not easily readable or mergeable like scripting text files. So we had to provide a tool to prevent unnecessary blocking of the files, be able to track differences across versions

It worked in a similar manner you would expect from any similar text diff/merge tool would. Here you can see a screenshot from a diff view. There are options to show two or one way, additional options for navigation in the canvas and coloring to help identify the type of changes. (Like the (NEXT) blue frame here meaning a new node was added and the yellow frame indicating a change in the child count value.

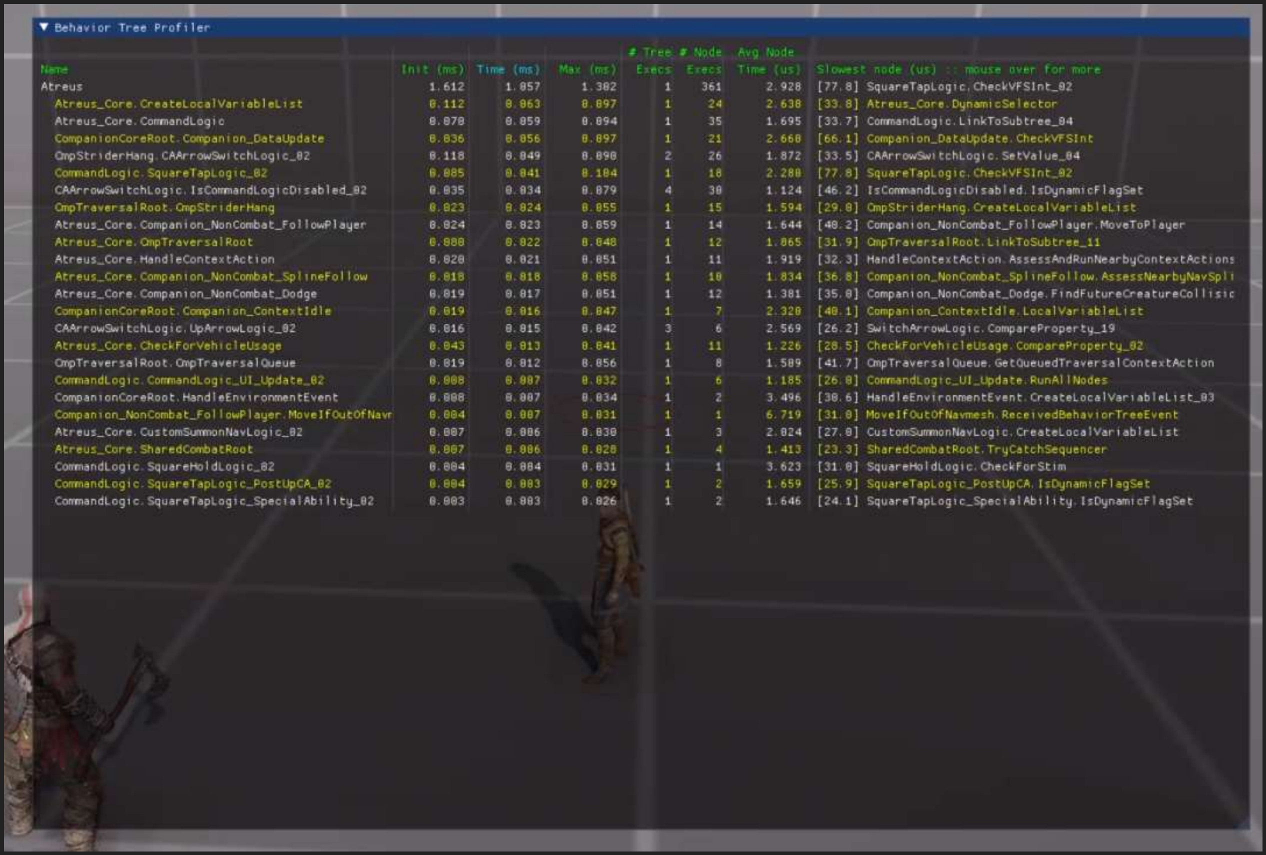
Same color were also respected in this list view (NEXT), which also gave you the option to click and jump to that change in canvas. You would also get a more detailed view of the values in any specific node on the attribute view window you can see here on left and right edges

We also supported filtering of the changes (NEXT) as seen here, to help fasten up navigating through many changes since even a positional changes on a node were being tracked. We had full perforce integration with diffing different versions of files.

Perforce was setup to use this tool for merging and notify the user on a successful merge, as well as give the user the option to review. When conflicts happened, it opened the file in merge view, which had a similar look and options to diff view here.

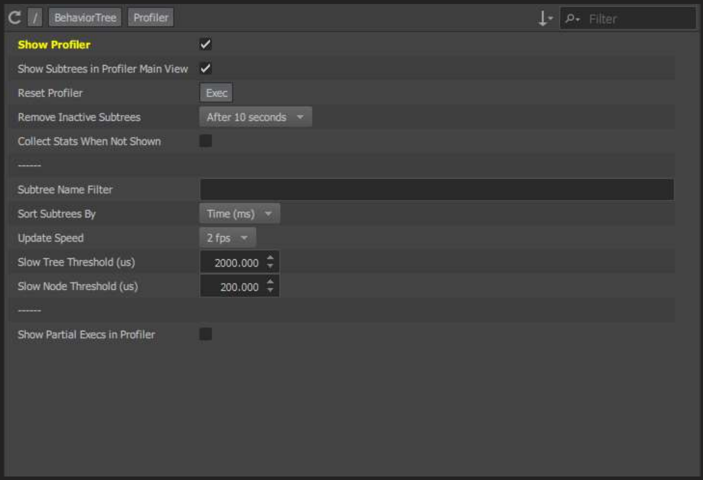
The merge worked successfully most of the time except for cases like different nodes being connected to same between revisions. Another bigger challenge appeared when we started working across multiple branches in perforce and were only integrating selected changelists and we could not guarantee sequential changes on the file. The conflict resolution view became way more important then

Performance Tool



▼ Behavior Tree Profiler

				# Tree	# Node	Avg Node	
Name	Init (ms)	Time (ms)	Max (ms)	Execs	Execs	Time (us)	Slowest node (us) :: mouse over for more
Atreus	1.612	1.057	1.382	1	361	2.928	[77.8] SquareTapLogic.CheckVFSInt_02
Atreus_Core.CreateLocalVariableList	0.112	0.063	0.097	1	24	2.638	[33.8] Atreus_Core.DynamicSelector
Atreus_Core.CommandLogic	0.078	0.059	0.094	1	35	1.695	[33.7] CommandLogic.LinkToSubtree_04
CompanionCoreRoot.Companion_DataUpdate	0.036	0.056	0.097	1	21	2.668	[66.1] Companion_DataUpdate.CheckVFSInt
OmpStriderHang.CAArrowSwitchLogic_02	0.118	0.049	0.090	2	26	1.872	[33.5] CAArrowSwitchLogic.SetValue_04
CommandLogic.SquareTapLogic_02	0.085	0.041	0.104	1	18	2.288	[77.8] SquareTapLogic.CheckVFSInt_02
CAArrowSwitchLogic.IsCommandLogicDisabled_02	0.035	0.034	0.079	4	30	1.124	[46.2] IsCommandLogicDisabled.IsDynamicFlagSet
OmpTraversalRoot.OmpStriderHang	0.023	0.024	0.055	1	15	1.594	[29.0] OmpStriderHang.CreateLocalVariableList
Atreus_Core.Companion_NonCombat_FollowPlayer	0.024	0.023	0.059	1	14	1.644	[40.2] Companion_NonCombat_FollowPlayer.MoveToPlayer
Atreus_Core.OmpTraversalRoot	0.080	0.022	0.048	1	12	1.865	[31.9] OmpTraversalRoot.LinkToSubtree_11
Atreus_Core.HandleContextAction	0.020	0.021	0.051	1	11	1.919	[32.3] HandleContextAction.AssessAndRunNearbyContextActions
Atreus_Core.Companion_NonCombat_SplineFollow	0.018	0.018	0.058	1	10	1.834	[36.8] Companion_NonCombat_SplineFollow.AssessNearbyNavSpli
Atreus_Core.Companion_NonCombat_Dodge	0.019	0.017	0.051	1	12	1.381	[35.0] Companion_NonCombat_Dodge.FindFutureCreatureCollisic
CompanionCoreRoot.Companion_ContextIdle	0.019	0.016	0.047	1	7	2.320	[40.1] Companion_ContextIdle.LocalVariableList
CAArrowSwitchLogic.UpArrowLogic_02	0.016	0.015	0.042	3	6	2.569	[26.2] SwitchArrowLogic.CompareProperty_19
Atreus_Core.CheckForVehicleUsage	0.043	0.013	0.041	1	11	1.226	[28.5] CheckForVehicleUsage.CompareProperty_02
OmpTraversalRoot.OmpTraversalQueue	0.019	0.012	0.056	1	8	1.589	[41.7] OmpTraversalQueue.GetQueuedTraversalContextAction
CommandLogic.CommandLogic_UI_Update_02	0.008	0.007	0.032	1	6	1.185	[26.0] CommandLogic_UI_Update.RunAllNodes
CompanionCoreRoot.HandleEnvironmentEvent	0.008	0.007	0.034	1	2	3.496	[30.6] HandleEnvironmentEvent.CreateLocalVariableList_03
Companion_NonCombat_FollowPlayer.MoveIfOutOfNavr	0.004	0.007	0.031	1	1	6.719	[31.0] MoveIfOutOfNavmesh.ReceivedBehaviorTreeEvent
Atreus_Core.CustomSummonNavLogic_02	0.007	0.006	0.030	1	3	2.024	[27.0] CustomSummonNavLogic.CreateLocalVariableList
Atreus_Core.SharedCombatRoot	0.007	0.006	0.028	1	4	1.413	[23.3] SharedCombatRoot.TryCatchSequencer
CommandLogic.SquareHoldLogic_02	0.004	0.004	0.031	1	1	3.623	[31.0] SquareHoldLogic.CheckForStim
CommandLogic.SquareTapLogic_PostUpCA_02	0.004	0.003	0.029	1	2	1.659	[25.9] SquareTapLogic_PostUpCA.IsDynamicFlagSet
CommandLogic.SquareTapLogic_SpecialAbility_02	0.003	0.003	0.026	1	2	1.646	[24.1] SquareTapLogic_SpecialAbility.IsDynamicFlagSet



Show Profiler	☑
Show Subtrees in Profiler Main View	☑
Reset Profiler	Exec
Remove Inactive Subtrees	After 10 seconds
Collect Stats When Not Shown	□

Subtree Name Filter	
Sort Subtrees By	Time (ms)
Update Speed	2 fps
Slow Tree Threshold (us)	2000.000
Slow Node Threshold (us)	200.000

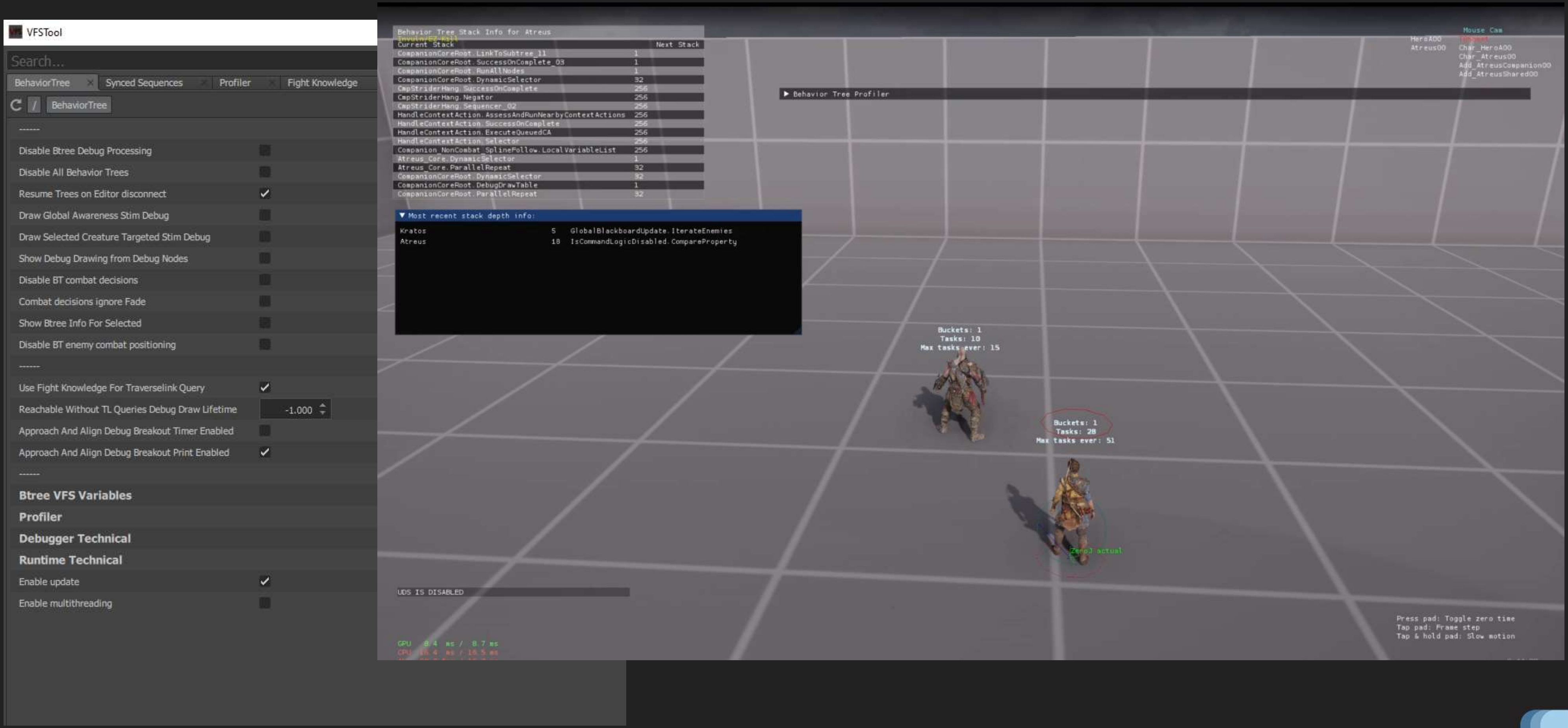
Show Partial Execs in Profiler	□

Goal : have an easy in game method to profile specific btrees. Easy for programmers but also accessible for designers

- It came in with some additional options like (point to new image)

Helped give an easy way for design to pinpoint the problem nodes or problem callstacks within the tree
For programming – can get less handy when you get to the point of needing to squeeze every bit of performance late production so not a replacement for regular profiling tools

Runtime Debug Options



VFSTool

Search...

- Disable Btree Debug Processing □
- Disable All Behavior Trees □
- Resume Trees on Editor disconnect ☑
- Draw Global Awareness Stim Debug □
- Draw Selected Creature Targeted Stim Debug □
- Show Debug Drawing from Debug Nodes □
- Disable BT combat decisions □
- Combat decisions ignore Fade □
- Show Btree Info For Selected □
- Disable BT enemy combat positioning □

- Use Fight Knowledge For Traverselink Query ☑
- Reachable Without TL Queries Debug Draw Lifetime -1.000
- Approach And Align Debug Breakout Timer Enabled □
- Approach And Align Debug Breakout Print Enabled ☑

Btree VFS Variables

Profiler

Debugger Technical

Runtime Technical

- Enable update ☑
- Enable multithreading □

```
Behavior Tree Stack Info for Atreus
Current Stack                                                  Next Stack
CompanionCoreRoot.LinkToSubtree_11                       1
CompanionCoreRoot.SuccessOnComplete_03                   1
CompanionCoreRoot.RunAllNodes                            1
CompanionCoreRoot.DynamicSelector                        32
CmpStriderHang.SuccessOnComplete                         256
CmpStriderHang.Negator                                   256
CmpStriderHang.Sequencer_02                              256
HandleContextAction.AssessAndRunNearbyContextActions     256
HandleContextAction.SuccessOnComplete                    256
HandleContextAction.ExecuteQueuedCA                      256
HandleContextAction.Selector                             256
Companion_NonCombat_SplineFollow.LocalVariableList       256
Atreus_Core.DynamicSelector                              1
Atreus_Core.ParallelRepeat                               32
CompanionCoreRoot.DynamicSelector                        32
CompanionCoreRoot.DebugDrawTable                         1
CompanionCoreRoot.ParallelRepeat                         32
```

```
▼ Most recent stack depth info:
Kratos        5    GlobalBlackboardUpdate.IterateEnemies
Atreus        18   IsCommandLogicDisabled.CompareProperty
```

```
▶ Behavior Tree Profiler
```

```
                Mouse Cam
HeroA00
Atreus00        Char_HeroA00
                Char_Atreus00
                Add_AtreusCompanion00
                Add_AtreusShared00
```

```
Buckets: 1
Tasks: 10
Max tasks ever: 15
```

```
Buckets: 1
Tasks: 28
Max tasks ever: 51
```

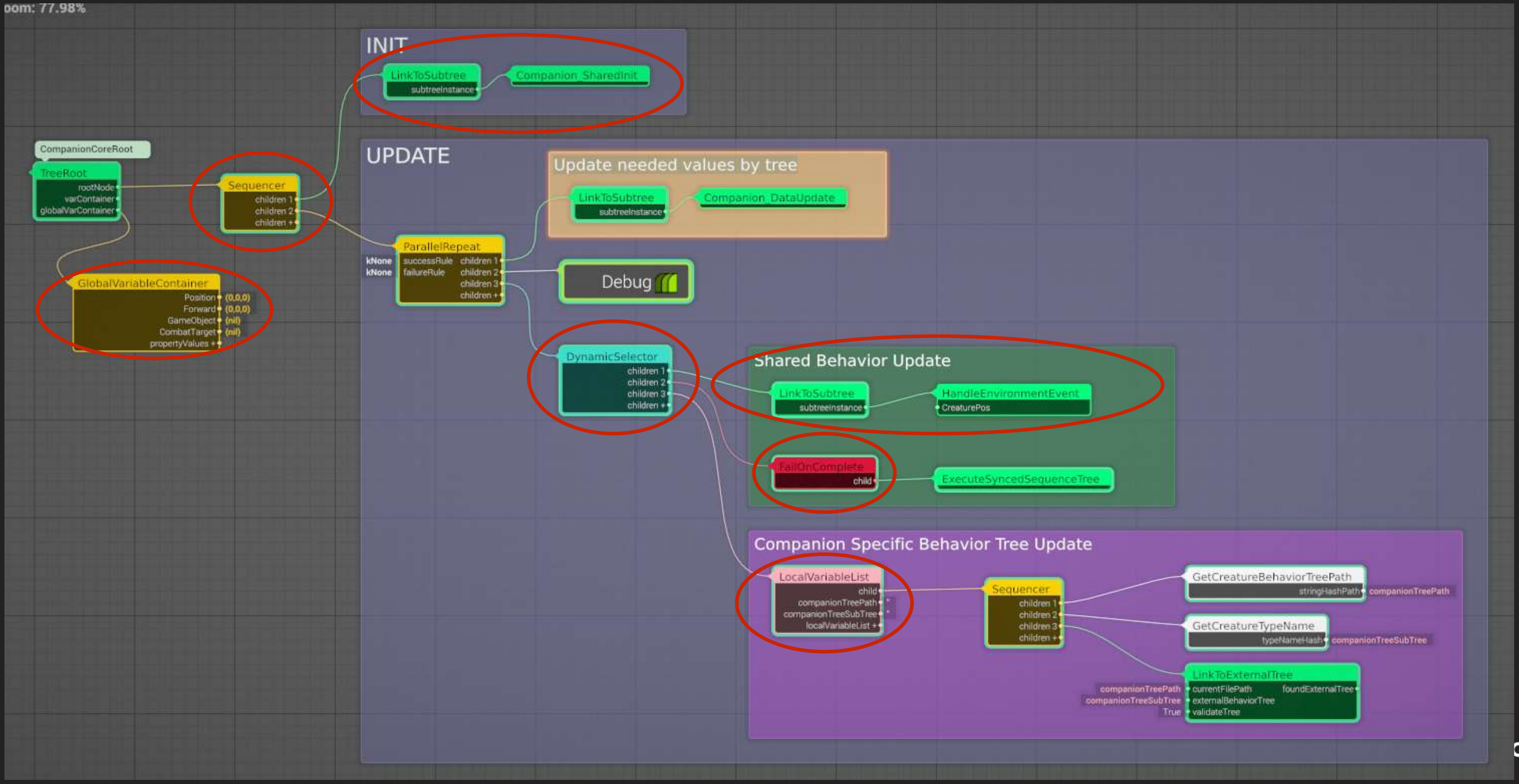
```
UDS IS DISABLED

GPU   8.4  ms /  8.7 ms
CPU  16.4  ms / 16.5 ms
```

```
Press pad: Toggle zero time
Tap pad: Frame step
Tap & hold pad: Slow motion
```

Santa Monica Studio

We also had many useful programming facing debug views in game. I will not go through each of them, but this is an example. This shows important things like current running node stack at the top left. And important info like max depth, max nodes and bucket info for behavior trees for creatures. This kind of in game easily accessibly debug options were not only useful for programmers but it was also fastened up the tracking of issues when other programmers, designers or QA ran into them since we could get a debug view right when it happened.

Behavior Trees

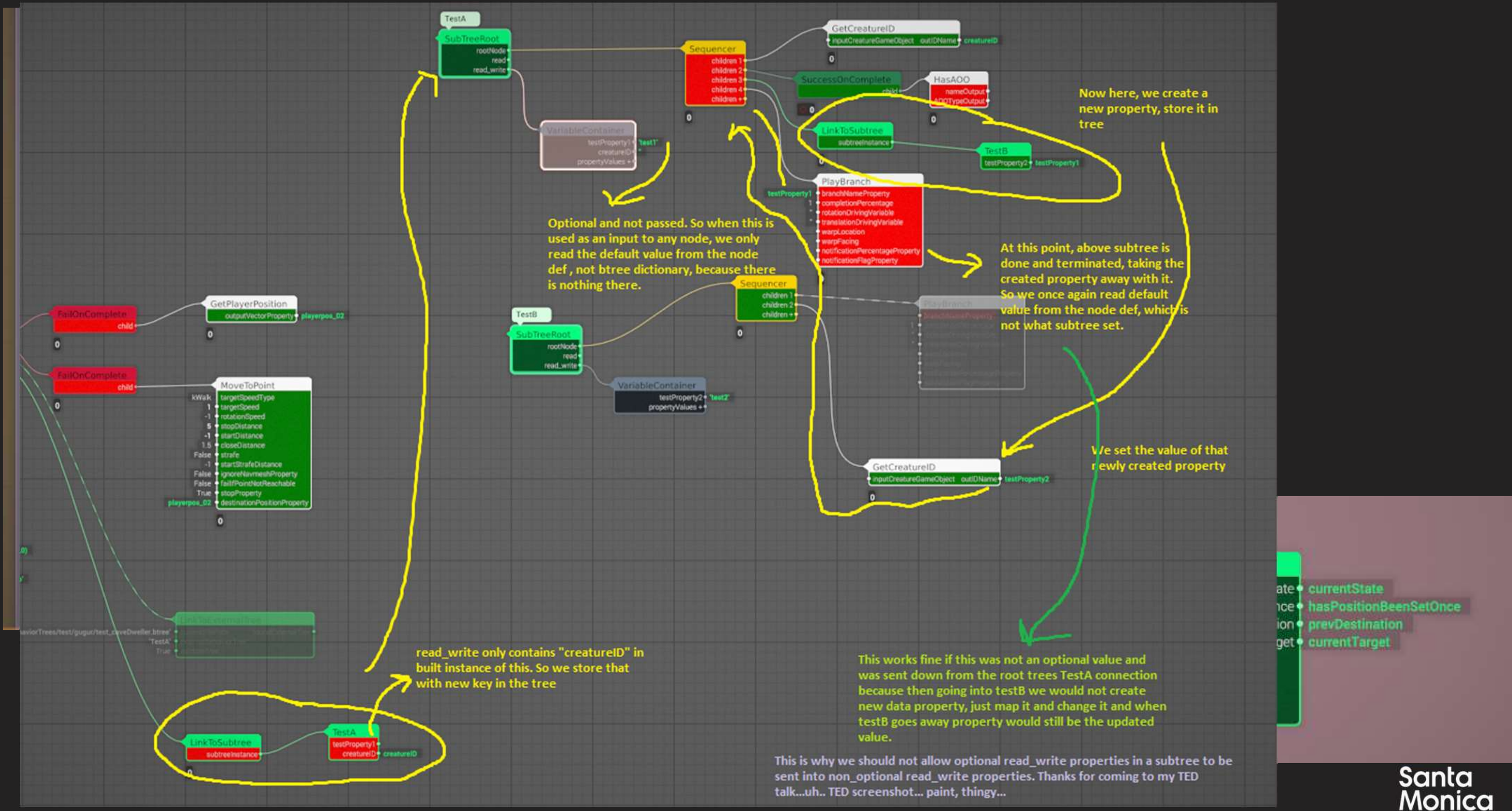


Now that we covered how we transitioned to behavior trees, lets actually talk a little bit about how we used them. I am not going to dive too much into how it works, but I would like to give a quick run down to serve as context for some upcoming info

What you see here is a simplified version of what a companion btree might look like.

- we have common nodes like– Dynamic selectors, sequencer, ParallelRepeater as well as some helpers like "FailOnComplete, SucessOnComplete, KeepRunning"
- You can also see these linking nodes that can call into another subtree that is defined elsewhere. So this acts very much like a function that can be called at any given point in the tree
- We also have these container nodes for variables. Global variables are accessible by any node in the tree and stores as a key-value pair in a runtime dictionary, using its namehash as key since we can guarantee it will be unique.
- Local variables are on the other hand are only accessibly within their scope by its children. But if you consider the same subtree can be injected multiple times at any given point in the tree, We cannot guarantee name itself will be unique in root tree level we create a different hash appending their name to their subtree call hierarchy.
- Additionally Subtrees can also have their own variables that can be passed by the caller

Behavior Trees



Optional and not passed. So when this is used as an input to any node, we only read the default value from the node def , not btree dictionary, because there is nothing there.

Now here, we create a new property, store it in tree

At this point, above subtree is done and terminated, taking the created property away with it. So we once again read default value from the node def, which is not what subtree set.

We set the value of that newly created property

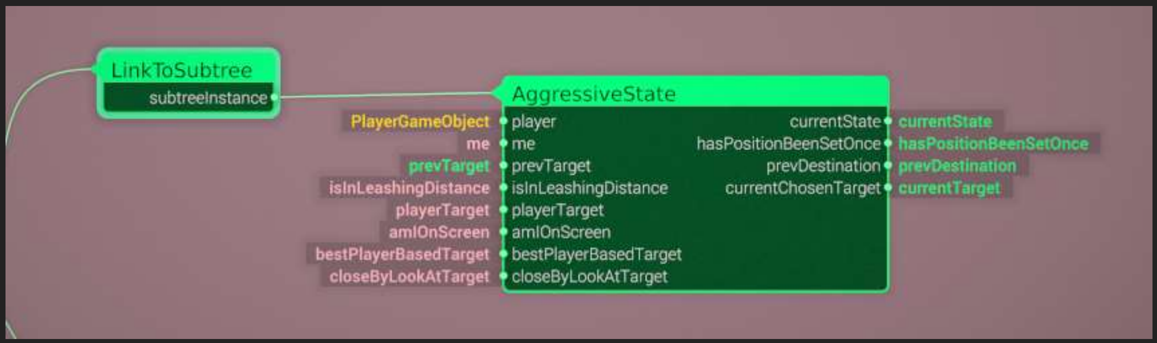
read_write only contains "creatureID" in built instance of this. So we store that with new key in the tree

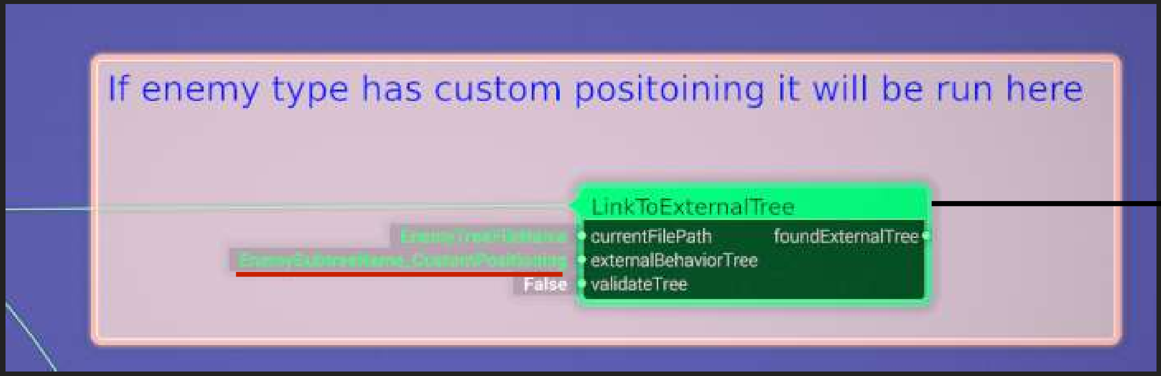
This works fine if this was not an optional value and was sent down from the root trees TestA connection because then going into testB we would not create new data property, just map it and change it and when testB goes away property would still be the updated value.

This is why we should not allow optional read_write properties in a subtree to be sent into non_optional read_write properties. Thanks for coming to my TED talk...uh.. TED screenshot... paint, thingy...

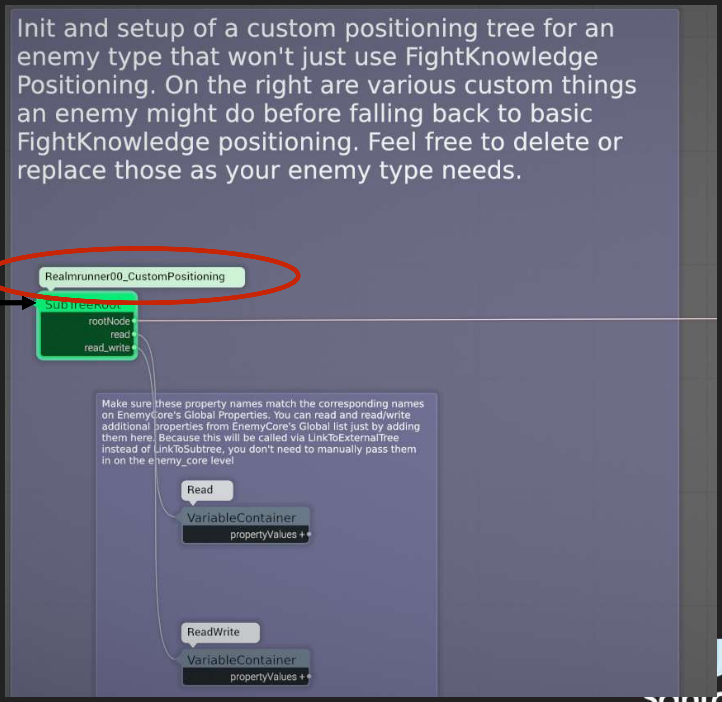
If you look at this example of a subtree of autonomous logic for companion, you can see another subtree call here :
player game object is a global variable of the root tree. Pink variables are local variables that are created by this subtree and green ones are passed in by the caller.
Also the variables on the left are considered inputs while the ones on the right are output
So all of this required some special treatment in the dictionary based on the unique node that makes the subtree connection.
It gets very complicated very quick, which led me to draw crazy flow diagrams when explaining what is going on under the hood during production because Handling these variables in runtime and validating data in the editor was crucial for our btrees and their stability

- So lets talk a little bit more about these linking nodes.
- We actually have 2 different ways





If enemy type has custom positoining it will be run here



Init and setup of a custom positioning tree for an enemy type that won't just use FightKnowledge Positioning. On the right are various custom things an enemy might do before falling back to basic FightKnowledge positioning. Feel free to delete or replace those as your enemy type needs.

Make sure these property names match the corresponding names on EnemyCore's Global Properties. You can read and read/write additional properties from EnemyCore's Global list just by adding them here. Because this will be called via LinkToExternalTree instead of LinkToSubtree, you don't need to manually pass them in on the enemy_core level

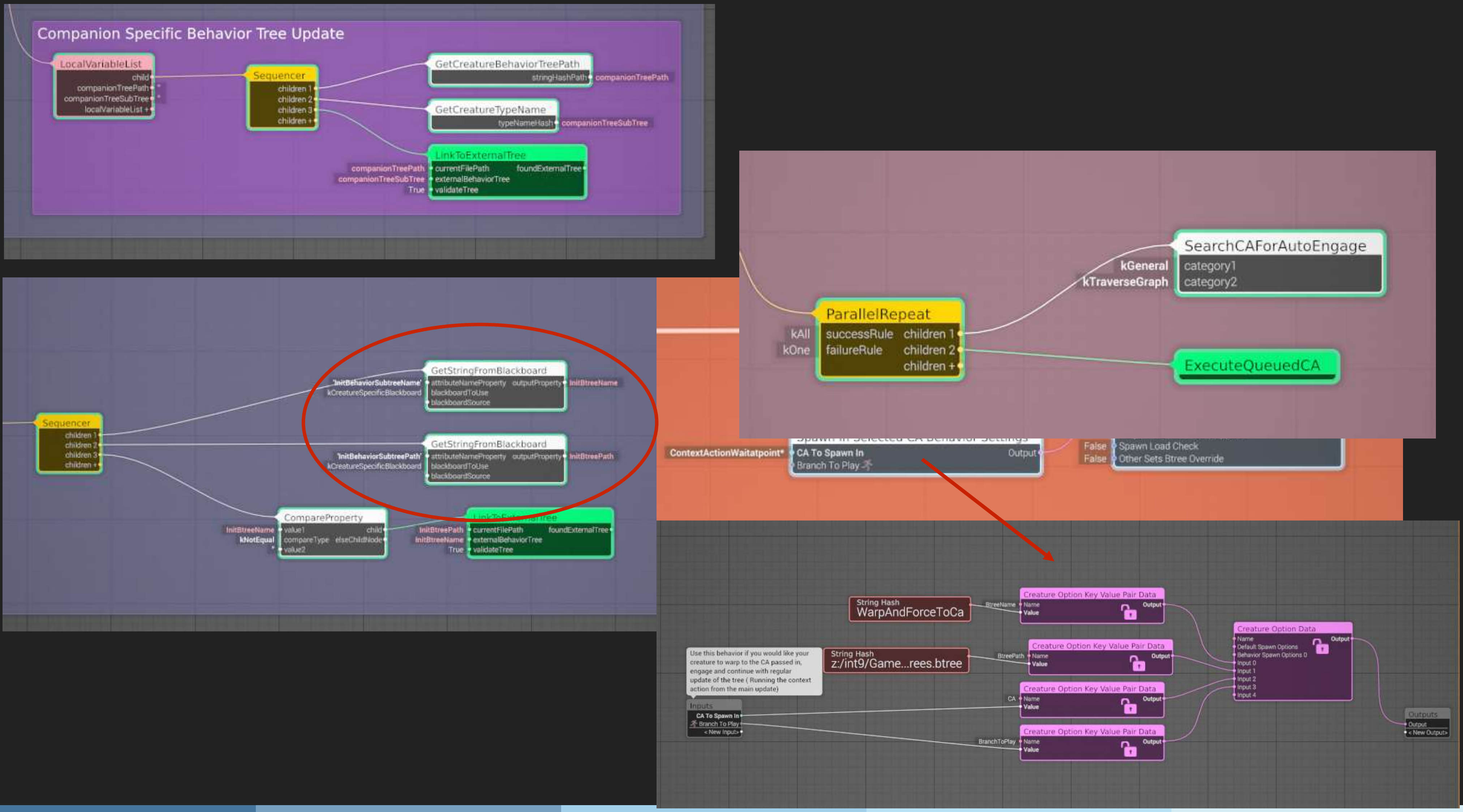
- When we know which subtree to call while authoring the tree we use LinkToSubtree nodes

  - Good for organization and shared functionality. Many utility subtrees like this one are used often by designers very much like a function call where each variable is passed in editing time.

- When the subtree we connect depends on on a runtime dynamic variable, we use LinkToExternalTree node. It requires the name of the subtree as well as the file path to it, which in this case are variables in the tree

  This example is from the core enemy behavior tree that every enemy runs, so we use this to find the specific creatures custom positioning logic.. If a realmRunner was currently running this behavior tree for example, it would connect to a subtree that looks like this. (IMAGE), as you can see it is named RealmRunner00_CustomPositioning. So designers are in charge of ensuring when they build the "enemySubtreeName_CustomPositioning" string there is a valid subtree to be called

  - While they are really Good for modularization while adding custom setup for variability
  - They are also more error prone as we can no longer validate in the editor



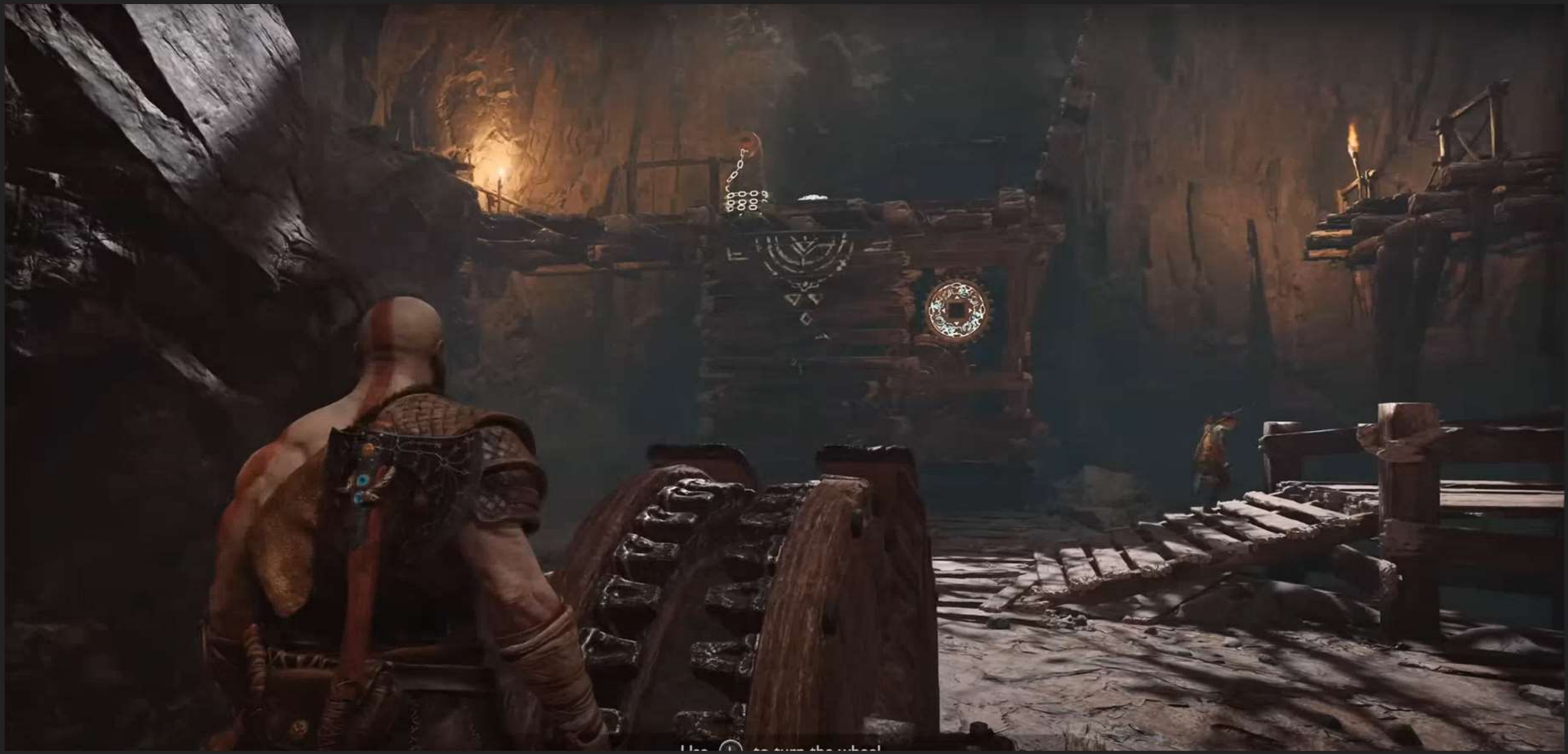
Use this behavior if you would like your creature to warp to the CA passed in, engage and continue with regular update of the tree (Running the context action from the main update)

As you have seen in enemy btree example, companions also used a similar setup to allow for custom logic while sharing a main behavior tree. In this example, you can also see how we might build the string for the subtree name.

But there were more usages

- Like Init behaviors : This is a logic that runs in init part of AI brain to run specific behavior that was set by the spawner in Level scripting. Each behavior sets the subtree name and path to be stored in blavkboard. It is is later read and passed to this this LinkToExternalTree node by behavior tree
- Additionally we had these derived nodes from LinkToExternalTree node that acts the same way but has specific ways of reading the subtree name on code
    - this ExecuteQueuedContextAction node for example reads the path/subtree name values from the Context Action System in game

You heard me talk about Context actions in the beginning and how we used them to achieve a lot of environment dependent behavior for AI. So lets take a closer look at them and we can get back to talking more about this one specific node

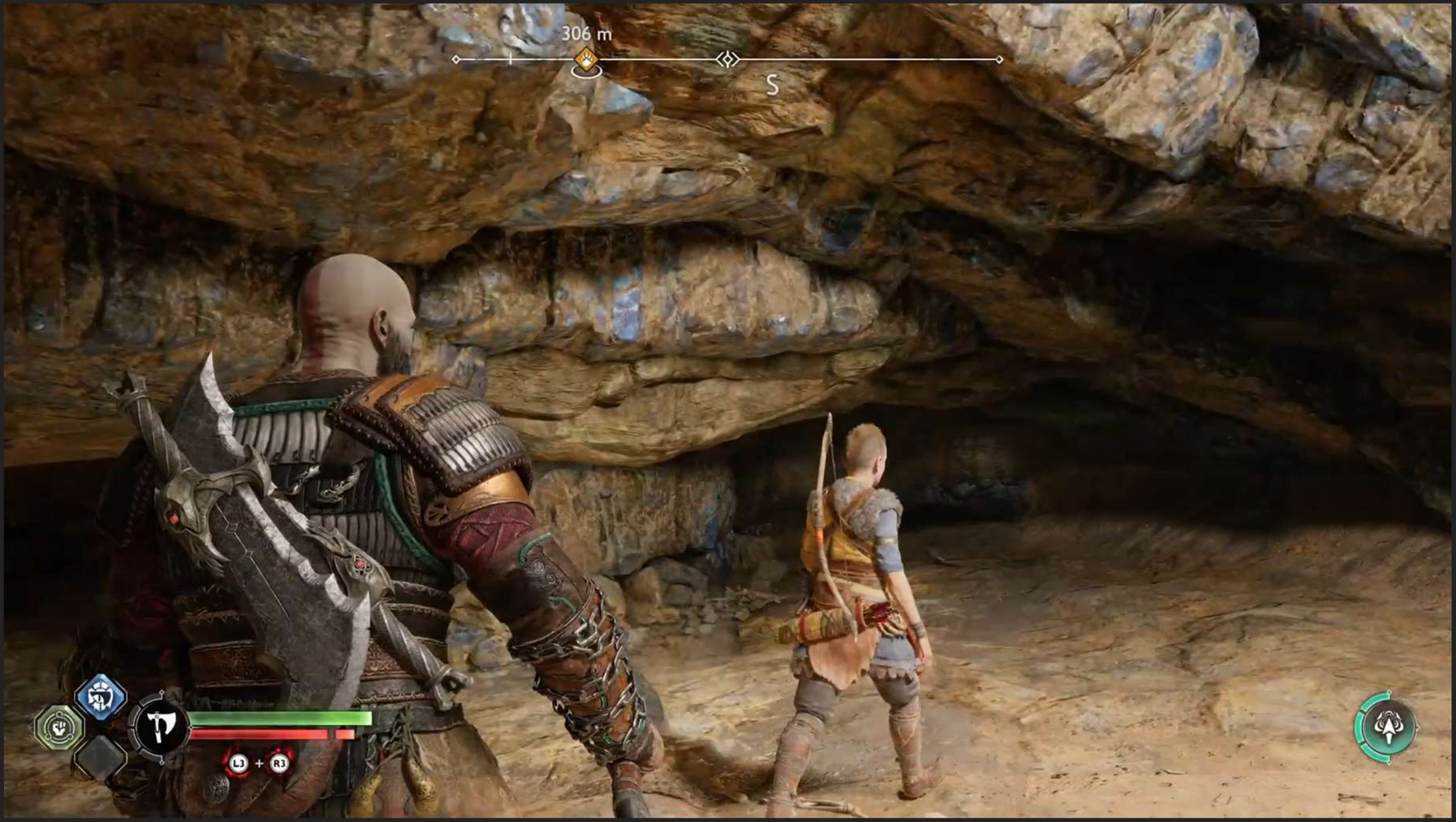


Context Actions

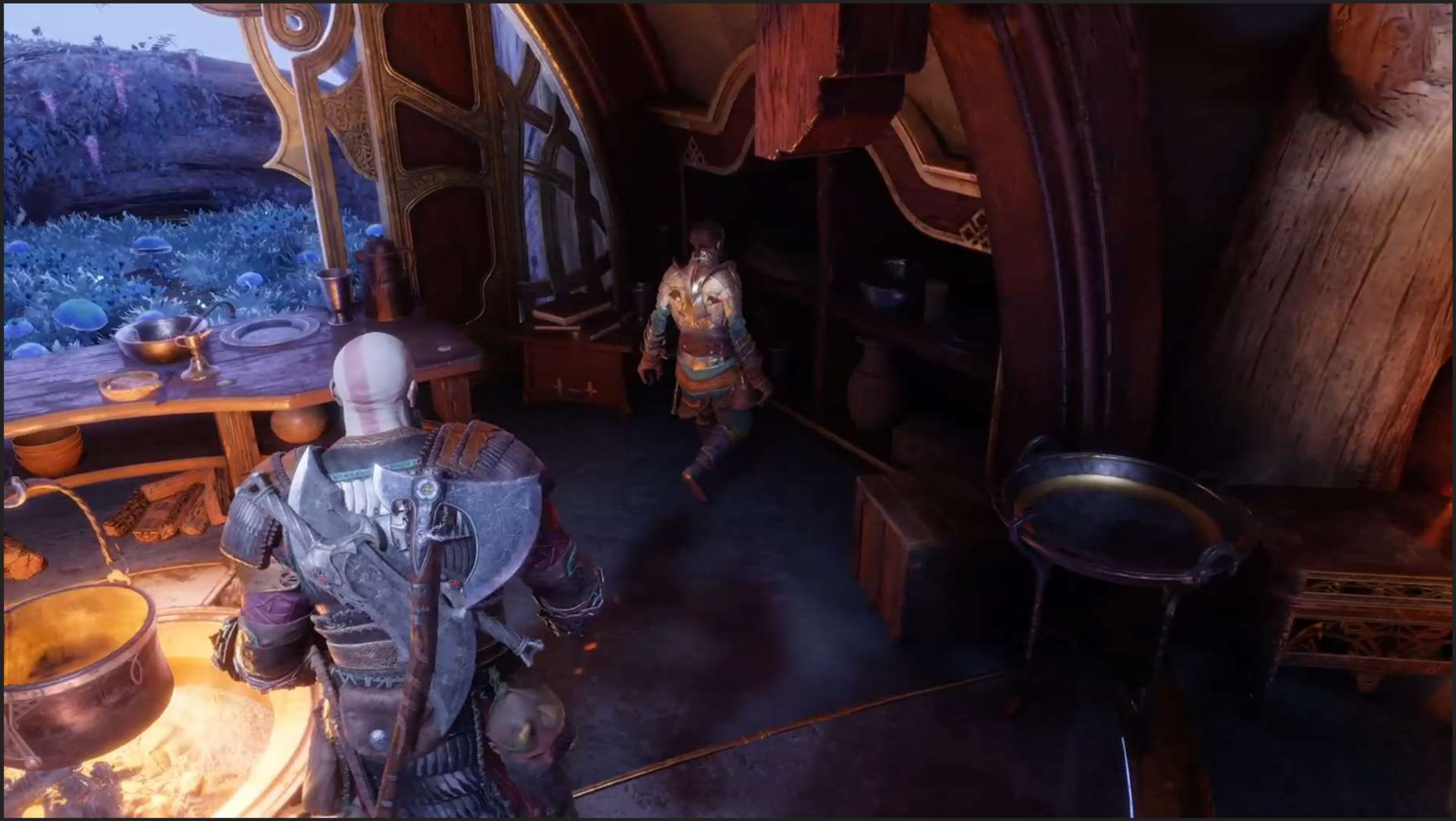
A set of injected behavior that is triggered at a specific location in the environment

Context Action system allows for designers to place an annotation in the world and assign it settings so that AI can engage with it and run a specific scoped behavior during the time of its engagement. They Usually consists all three entry, idle and an exit animations or any combination of them

This example is from GOW2018 young Atreus doing a "guide forward" based on a level beat in this area. So as a mentioned before we did have ways of doing this kind of behavior. But a big part of it was in ua, the code side was sprinkled across many different files and needed a heavy refactor to support some of the new requirements. We needed a system that can be more robust and can integrate with all other AI systems without much work needed to support many similar but variable AI behaviors that requires a world annotation
Some of the use cases in Ragnarok included

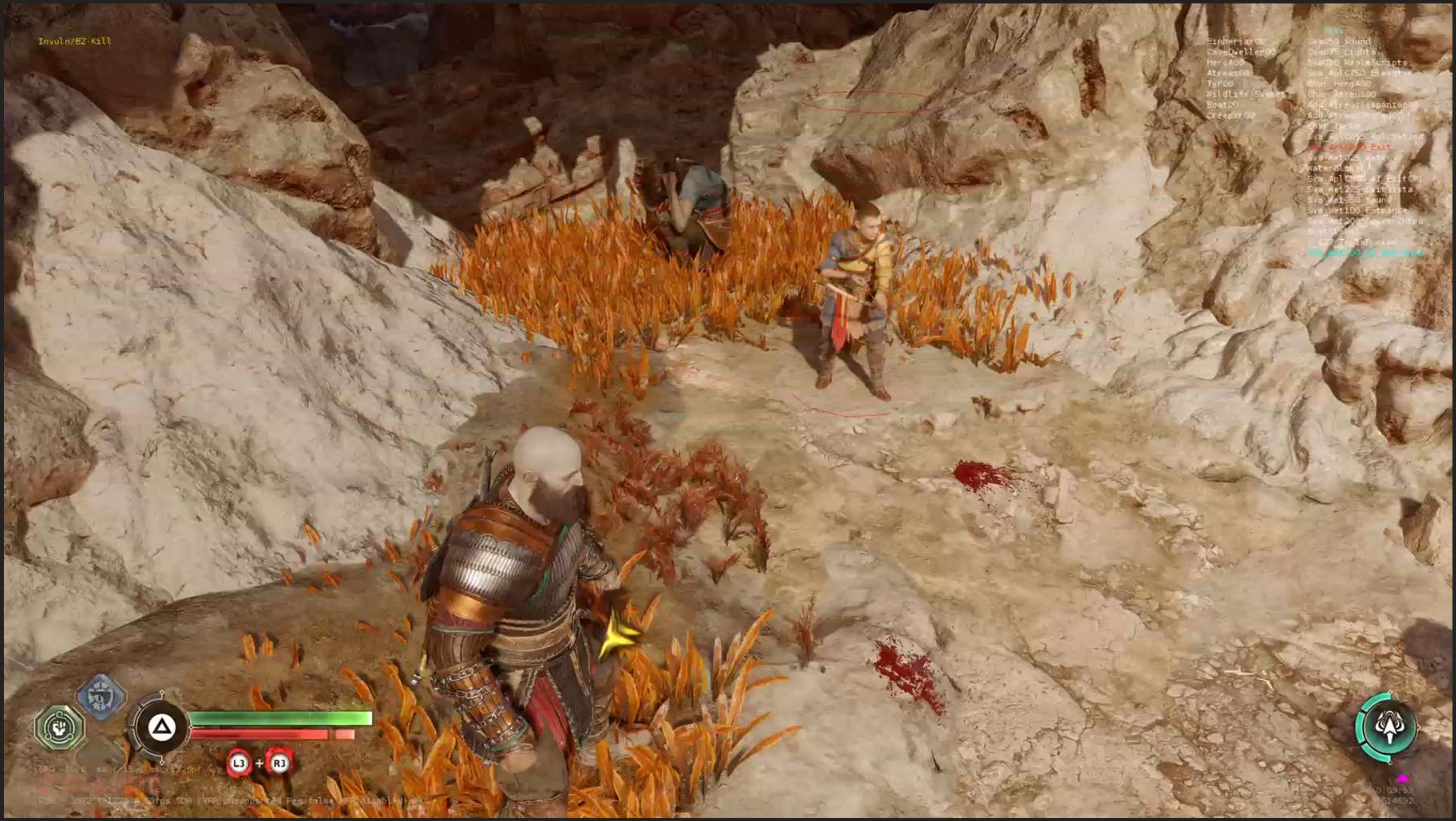


- Generic companion moments in levels to guide or hint player or to simply give them agency, believability. Things like point, crouch down, investigate were commonly placed in levels

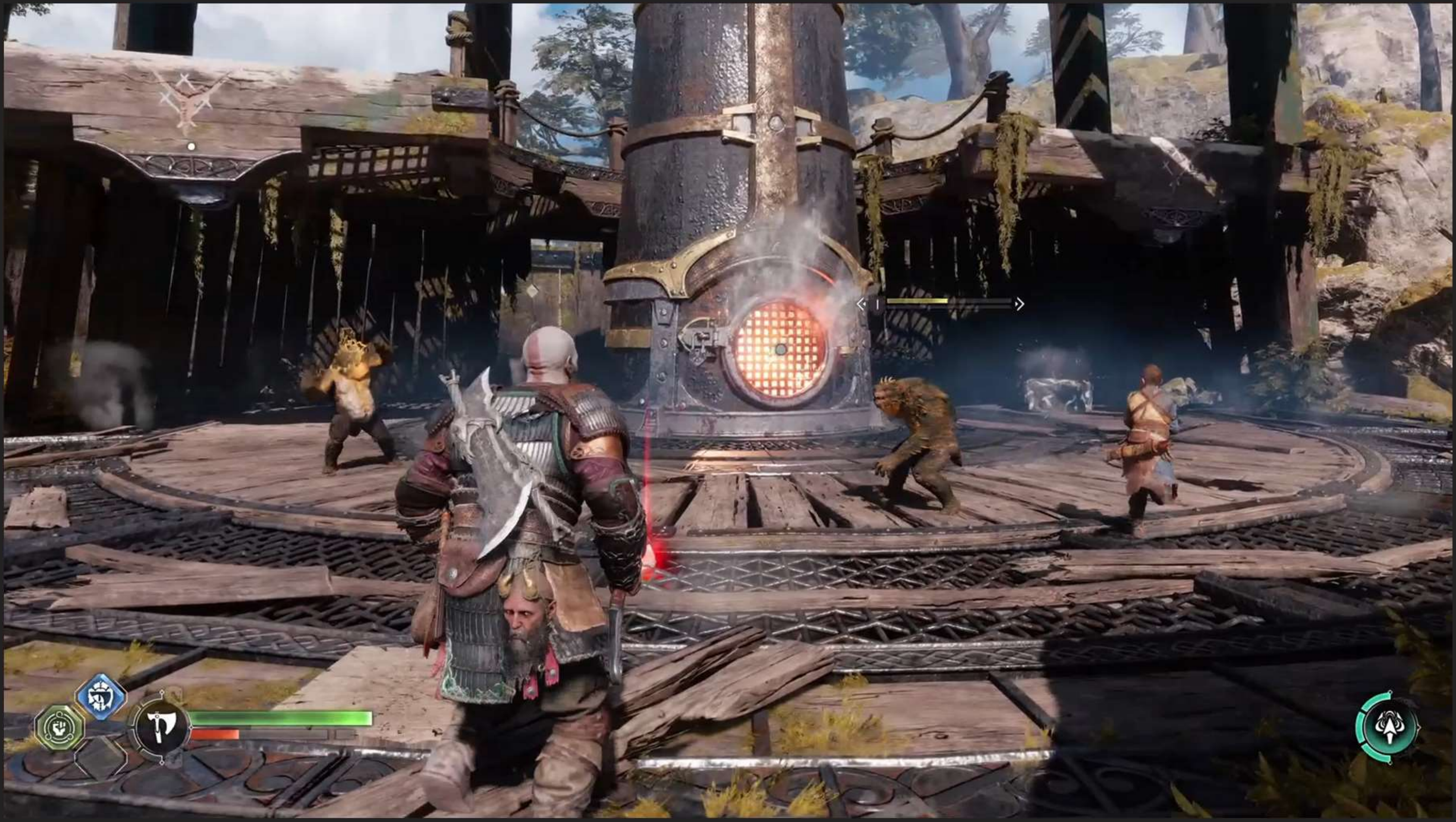


Contextual narrative moments. Here you can see Sindri looking for food for Tyr when he was at the house for the first time after being saved

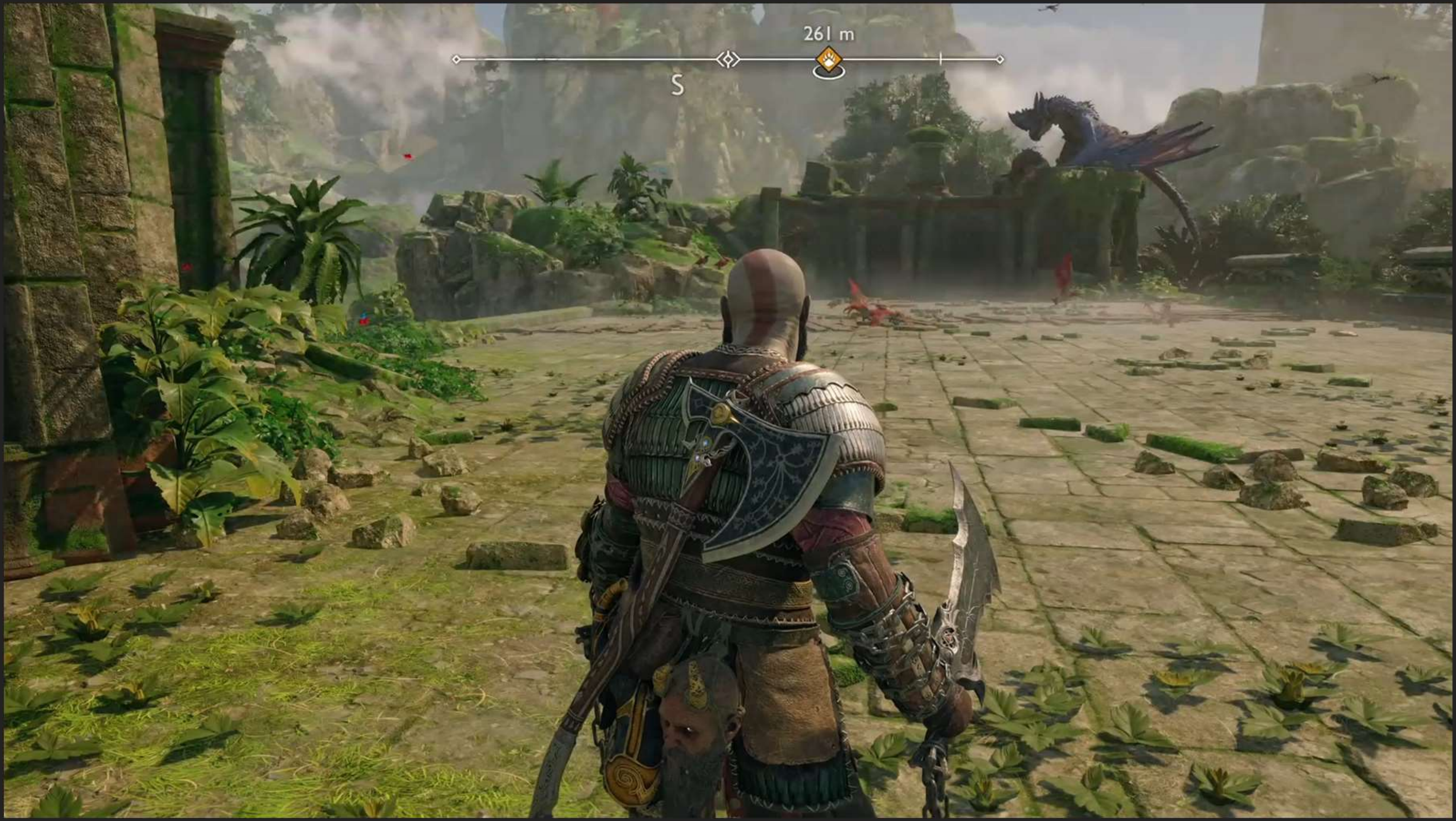
AI traversal behaviors
The logic of which path they follow, how far ahead they go, when they continue/pause is all in a context action subtree



We used them in combat. Here you can see both Atreus and Tyr in a context action during combat, Atreus specifically running a subtree to stand guard and shoot in place while Tyr (or Odin) cowardly hides away



And finally we used them for enhanced movement enemies



They also support multiple creatures! (which we did not really push much further this game but had very promising couple of uses cases like this one. A good start point for next project)

Context Action Setup Goals

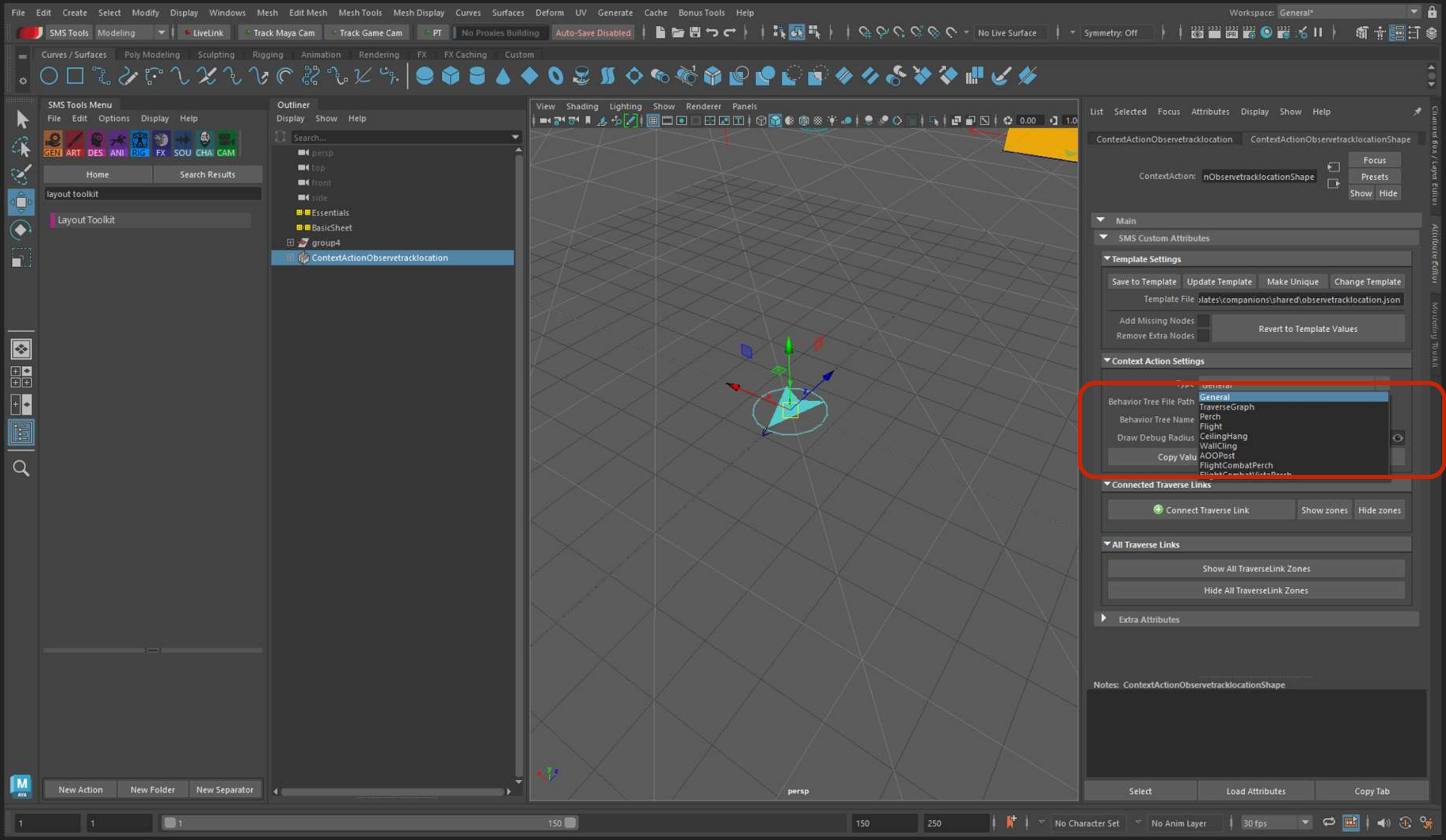
- Maximize modularity
- Allow for easy custom setup
- Max control for tech design teams
- Easy to use modules for designers

Santa Monica Studio

But how were the context actions themselves setup? Before we dive into the setup, I would like to talk about out goals setting these modules up

As you heard many times before, we wanted to maximize modularity, while supporting custom setups
We also needed to provide our Tech designers a lot of options to be able to create different modules without much programmer support
While Abstract the complication away for level/narrative/combat designers

Context Action Setup



ContextActionObservetracklocation

ContextActionObservetracklocationShape

ContextAction: nObservetracklocationShape

Template File plates\companions\shared\observetracklocation.json

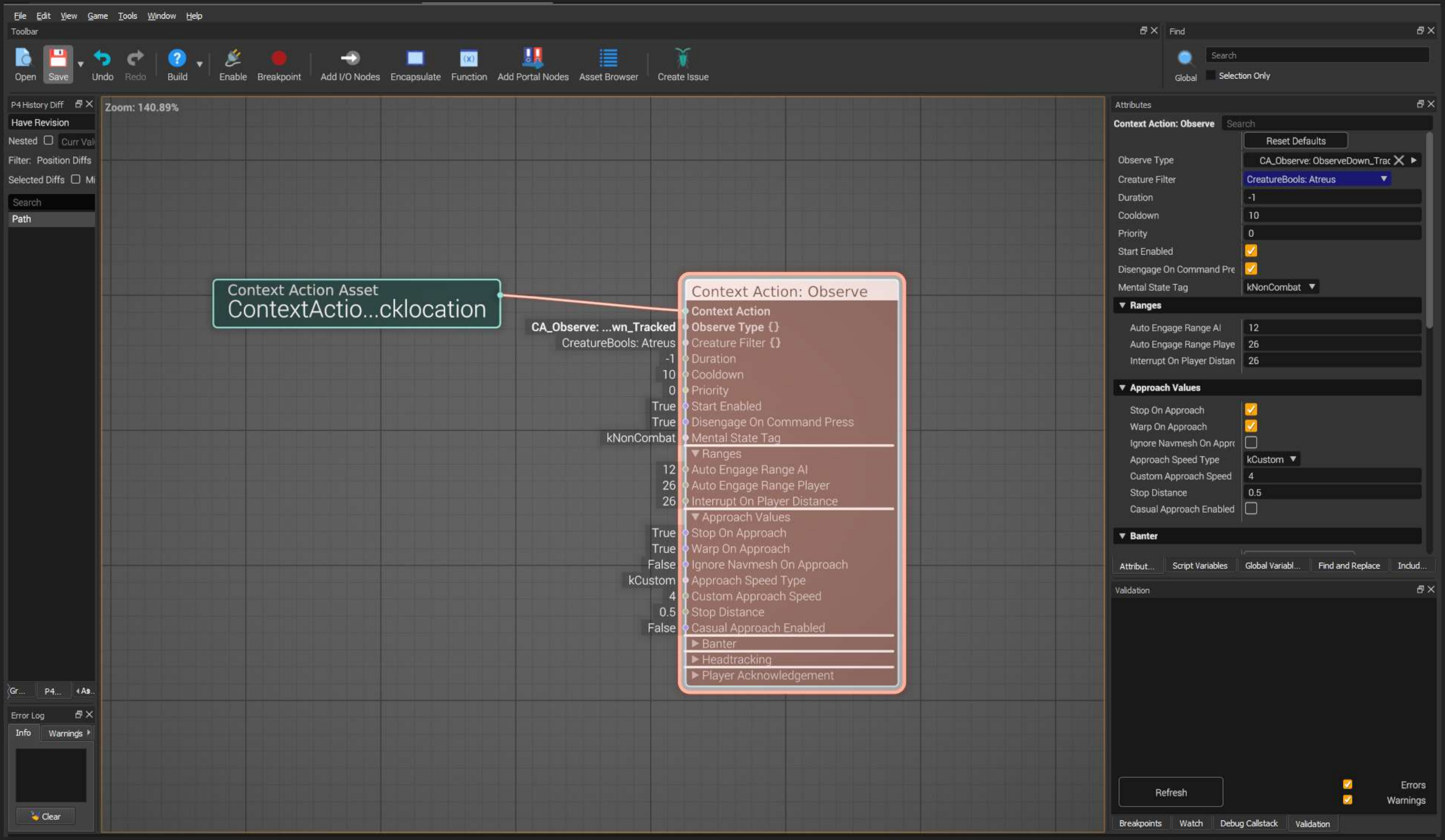
Type General

Notes: ContextActionObservetracklocationShape

Santa Monica Studio

In a very simple designer setup scenario, they would :
- Place markup in the level using one of our in house catalog tool (which autofills subtree name and path for them)
- Chose the category of context action

Context Action Setup



Context Action Asset
ContextActio...cklocation

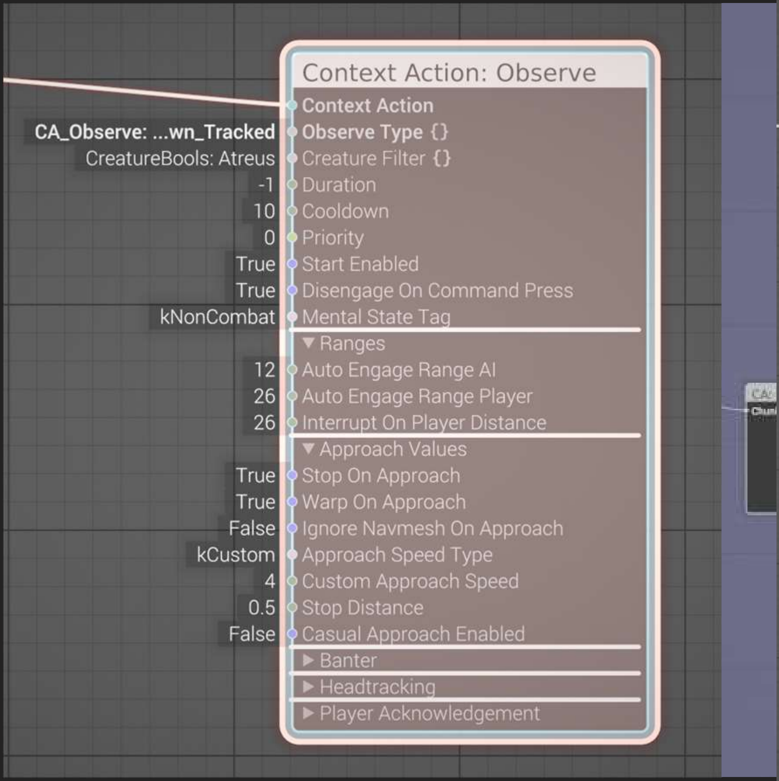
Context Action: Observe

	Context Action
CA_Observe: ...wn_Tracked	Observe Type {}
CreatureBools: Atreus	Creature Filter {}
-1	Duration
10	Cooldown
0	Priority
True	Start Enabled
True	Disengage On Command Press
kNonCombat	Mental State Tag
	▼ Ranges
12	Auto Engage Range AI
26	Auto Engage Range Player
26	Interrupt On Player Distance
	▼ Approach Values
True	Stop On Approach
True	Warp On Approach
False	Ignore Navmesh On Approach
kCustom	Approach Speed Type
4	Custom Approach Speed
0.5	Stop Distance
False	Casual Approach Enabled
	▶ Banter
	▶ Headtracking
	▶ Player Acknowledgement

Context Action: Observe

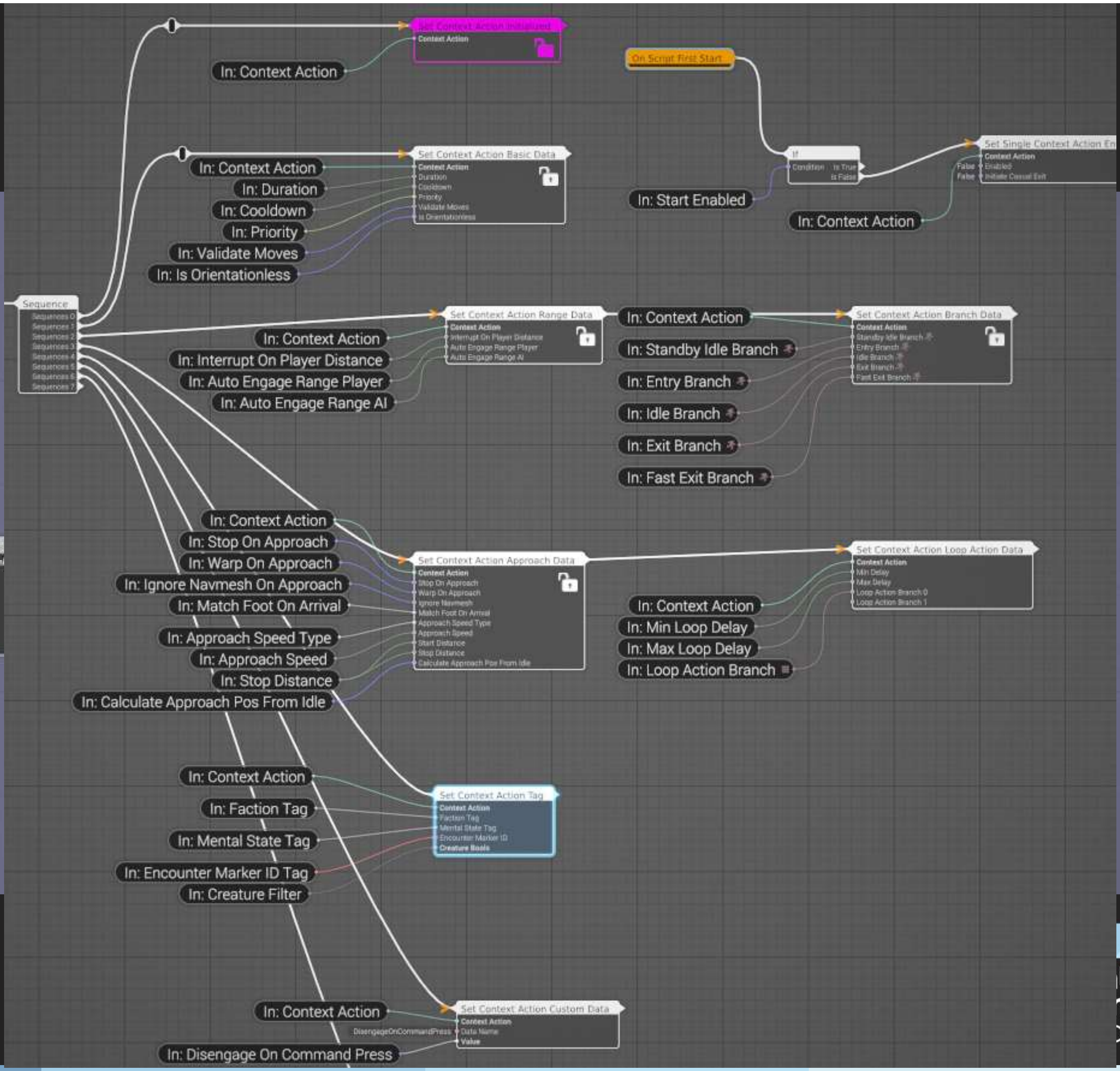
Observe Type	CA_Observe: ObserveDown_Trac
Creature Filter	CreatureBools: Atreus
Duration	-1
Cooldown	10
Priority	0
Start Enabled	☑
Disengage On Command Pre	☑
Mental State Tag	kNonCombat
▼ Ranges	
Auto Engage Range AI	12
Auto Engage Range Playe	26
Interrupt On Player Distan	26
▼ Approach Values	
Stop On Approach	☑
Warp On Approach	☑
Ignore Navmesh On Appr	□
Approach Speed Type	kCustom
Custom Approach Speed	4
Stop Distance	0.5
Casual Approach Enabled	□
▼ Banter	

- Then in level scripting, they can right click on that Context Action asset, and choose the type of context action that is associated with it in Visual Scripting
- Modify exposed values for that instance of that type
- You can see there are some toggle like duration of this action, range to engage in, specific approach parameters and more.



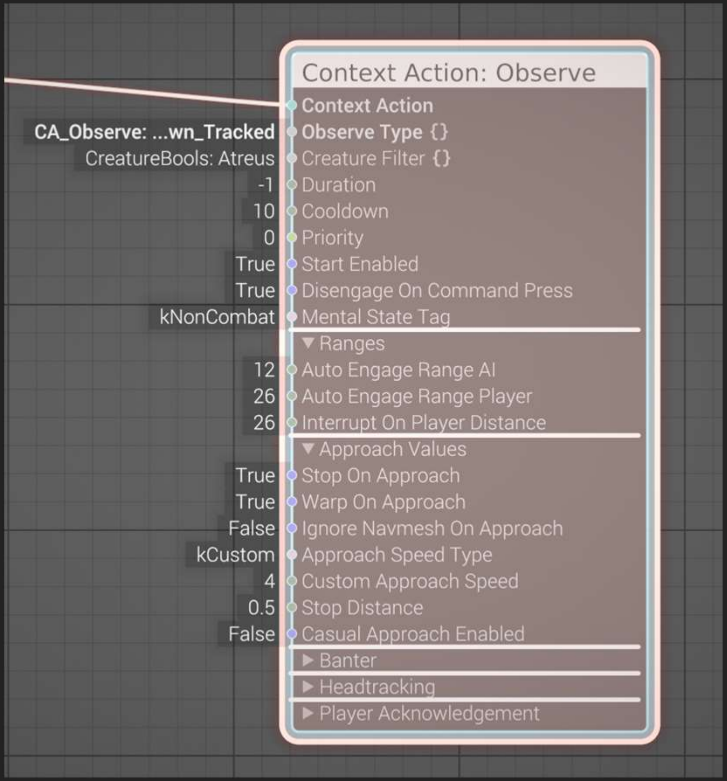
Context Action: Observe

	Context Action
CA_Observe: ...wn_Tracked	Observe Type {}
CreatureBools: Atreus	Creature Filter {}
-1	Duration
10	Cooldown
0	Priority
True	Start Enabled
True	Disengage On Command Press
kNonCombat	Mental State Tag
	▼ Ranges
12	Auto Engage Range AI
26	Auto Engage Range Player
26	Interrupt On Player Distance
	▼ Approach Values
True	Stop On Approach
True	Warp On Approach
False	Ignore Navmesh On Approach
kCustom	Approach Speed Type
4	Custom Approach Speed
0.5	Stop Distance
False	Casual Approach Enabled
	▶ Banter
	▶ Headtracking
	▶ Player Acknowledgement



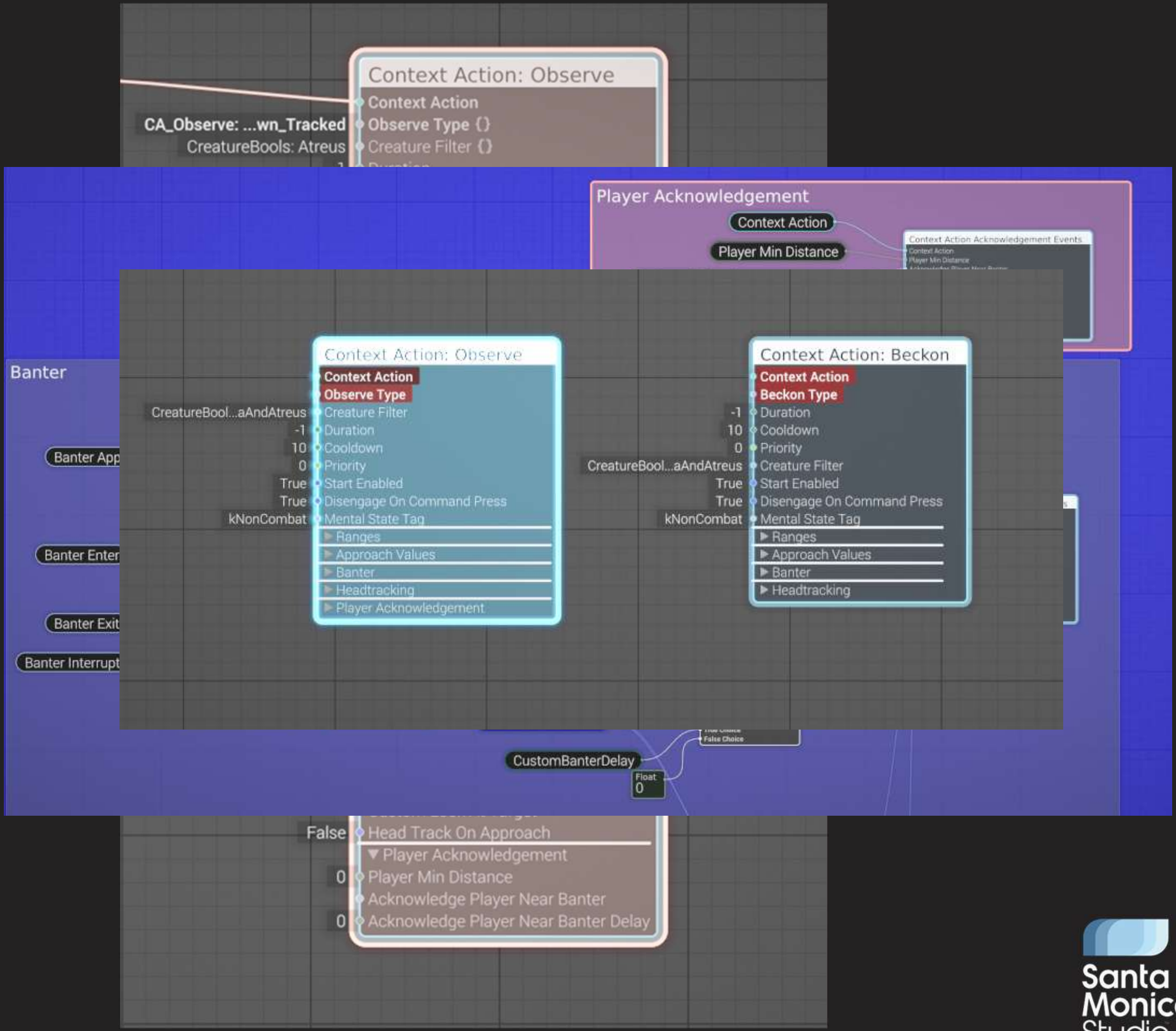
But you also realize there is nothing about enter, idle, exit animations we talked about.
Because that is hidden away within this embedded script. Each of these modules expands into more level scripting modules that both passes the values down but also sets non-changing values for that context action under the hood, like animation names for this observe context action that always remain same thus does not need to be exposed to user, which minimizes room for error
you can keep going many more levels down until you finally reach nodes that directly tie into game code.

This setup was not only good to minimize errors but it also made it very easy for us to tune default values and apply it to all existing instances and made bug fixes applicable globally



Context Action: Observe

Context Action
CA_Observe: ...wn_Tracked Observe Type {}
CreatureBools: Atreus Creature Filter {}
-1 Duration
10 Cooldown
0 Priority
True Start Enabled
True Disengage On Command Press
kNonCombat Mental State Tag
▼ Ranges
12 Auto Engage Range AI
26 Auto Engage Range Player
26 Interrupt On Player Distance
▼ Approach Values
True Stop On Approach
True Warp On Approach
False Ignore Navmesh On Approach
kCustom Approach Speed Type
4 Custom Approach Speed
0.5 Stop Distance
False Casual Approach Enabled
▶ Banter
▶ Headtracking
▶ Player Acknowledgement



Context Action: Observe

Context Action
CA_Observe: ...wn_Tracked Observe Type {}
CreatureBools: Atreus Creature Filter {}

Context Action: Observe

Context Action
Observe Type
CreatureBool...aAndAtreus Creature Filter
-1 Duration
10 Cooldown
0 Priority
True Start Enabled
True Disengage On Command Press
kNonCombat Mental State Tag
▶ Ranges
▶ Approach Values
▶ Banter
▶ Headtracking
▶ Player Acknowledgement

Context Action: Beckon

Context Action
Beckon Type
-1 Duration
10 Cooldown
0 Priority
CreatureBool...aAndAtreus Creature Filter
True Start Enabled
True Disengage On Command Press
kNonCombat Mental State Tag
▶ Ranges
▶ Approach Values
▶ Banter
▶ Headtracking

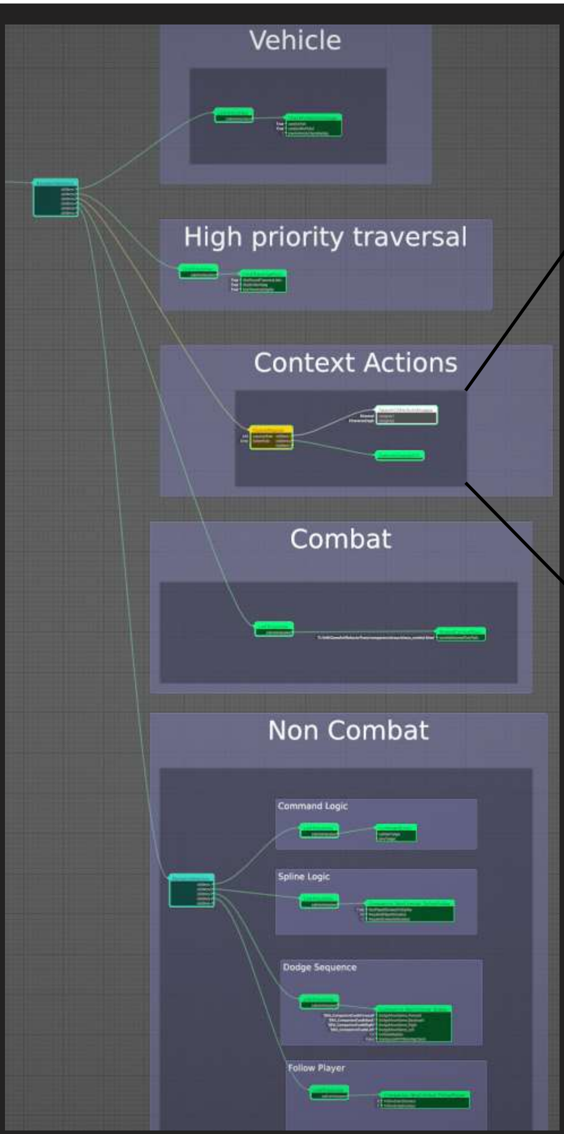
False Head Track On Approach
▼ Player Acknowledgement
0 Player Min Distance
Acknowledge Player Near Banter
0 Acknowledge Player Near Banter Delay

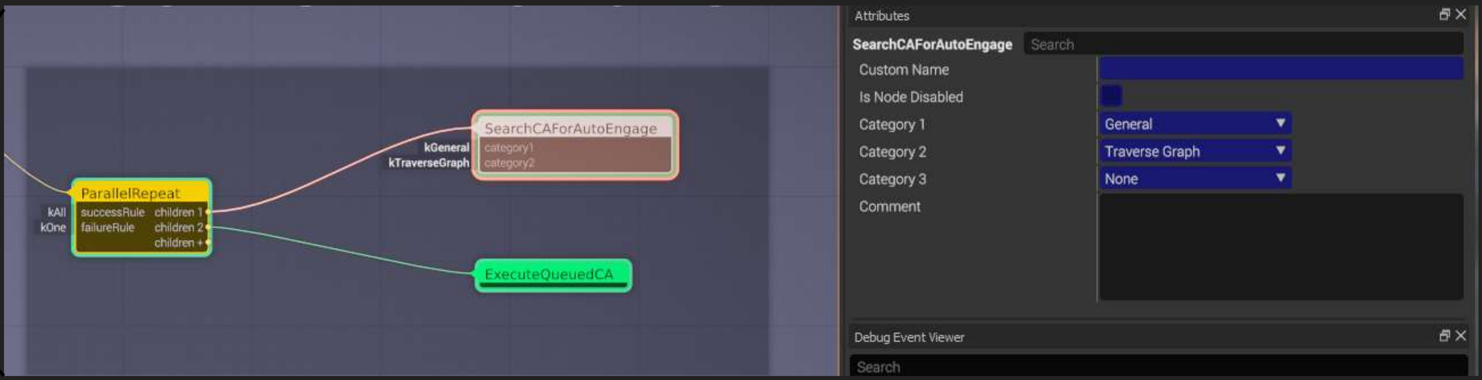
Santa Monica Studio

On top of that you also realize Context Action: Observe node comes in like this by default, the most commonly used value groups expanded, less likely ones are collapsed for ease of use. If we expand "Player acknowledgement" or "Banter" category, they come with their values and defaults. (goal of abstracting away complexity)

In this comparison view of these context actions you can also see some of these groups can easily be added or removed.
While some if these functionality was used by code and the behavior tree, some toggles like "player acknowledgeent" were actually just scripts independent from context action system, just utilizing exiting events like triggering a banter, only based on certain context action events.

All of these abilities together allowed us to easily share common functionality between different type of context actions while giving them an easy way to opt in or out of content as needed

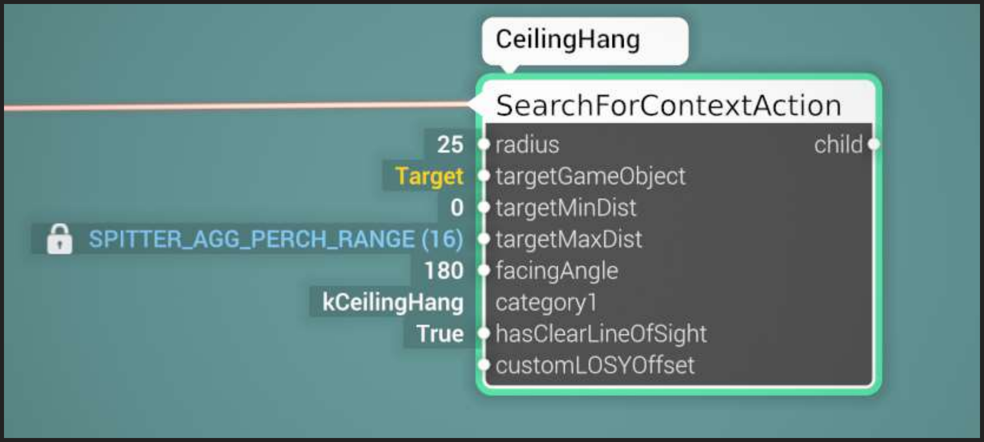




Attributes

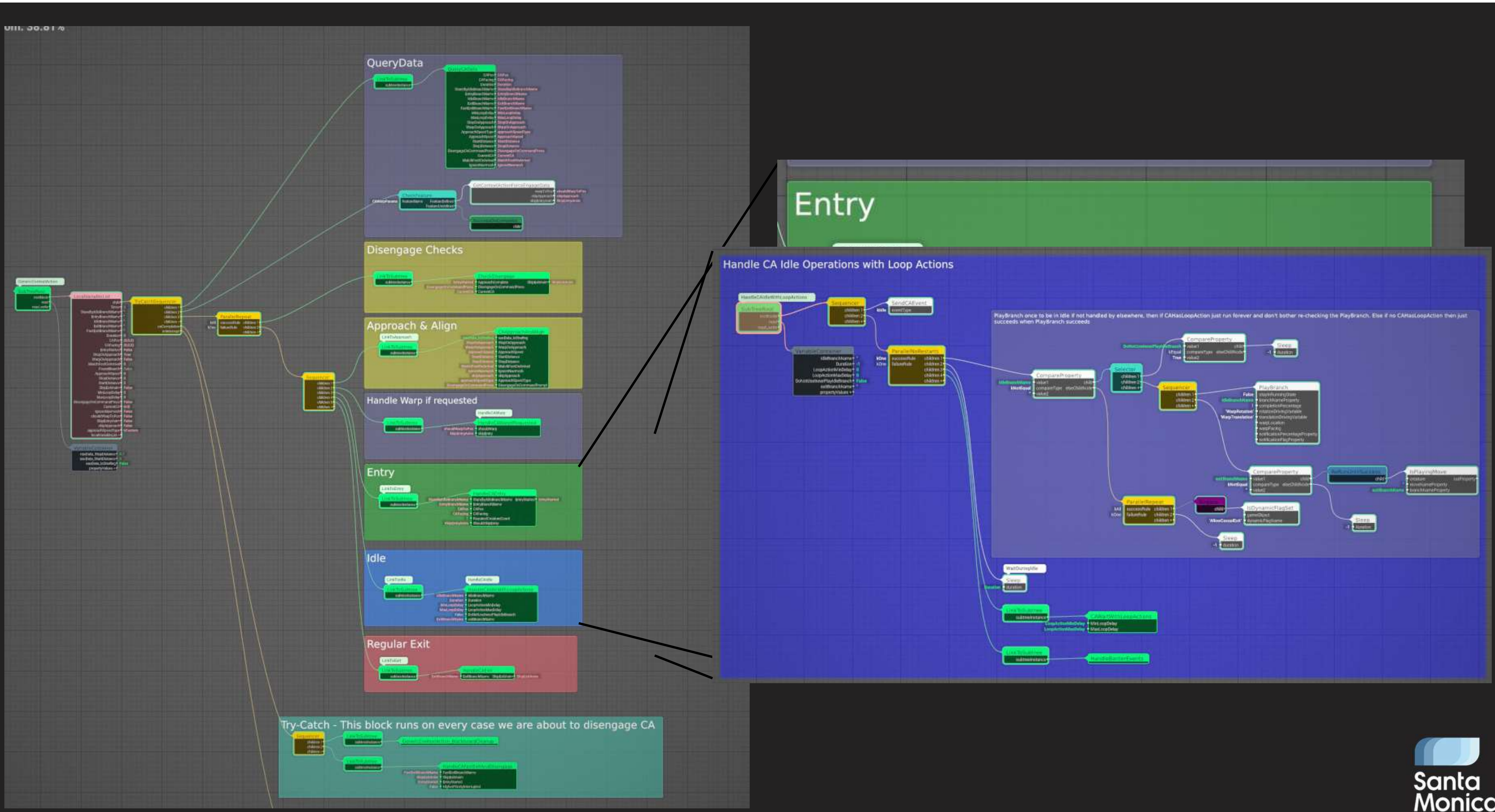
SearchCAForAutoEngage

Custom Name:
Is Node Disabled:
Category 1: General
Category 2: Traverse Graph
Category 3: None
Comment:



25 radius
Target targetGameObject
0 targetMinDist
SPITTER_AGG_PERCH_RANGE (16) targetMaxDist
180 facingAngle
kCeilingHang category1
True hasClearLineOfSight
customLOSYOffset

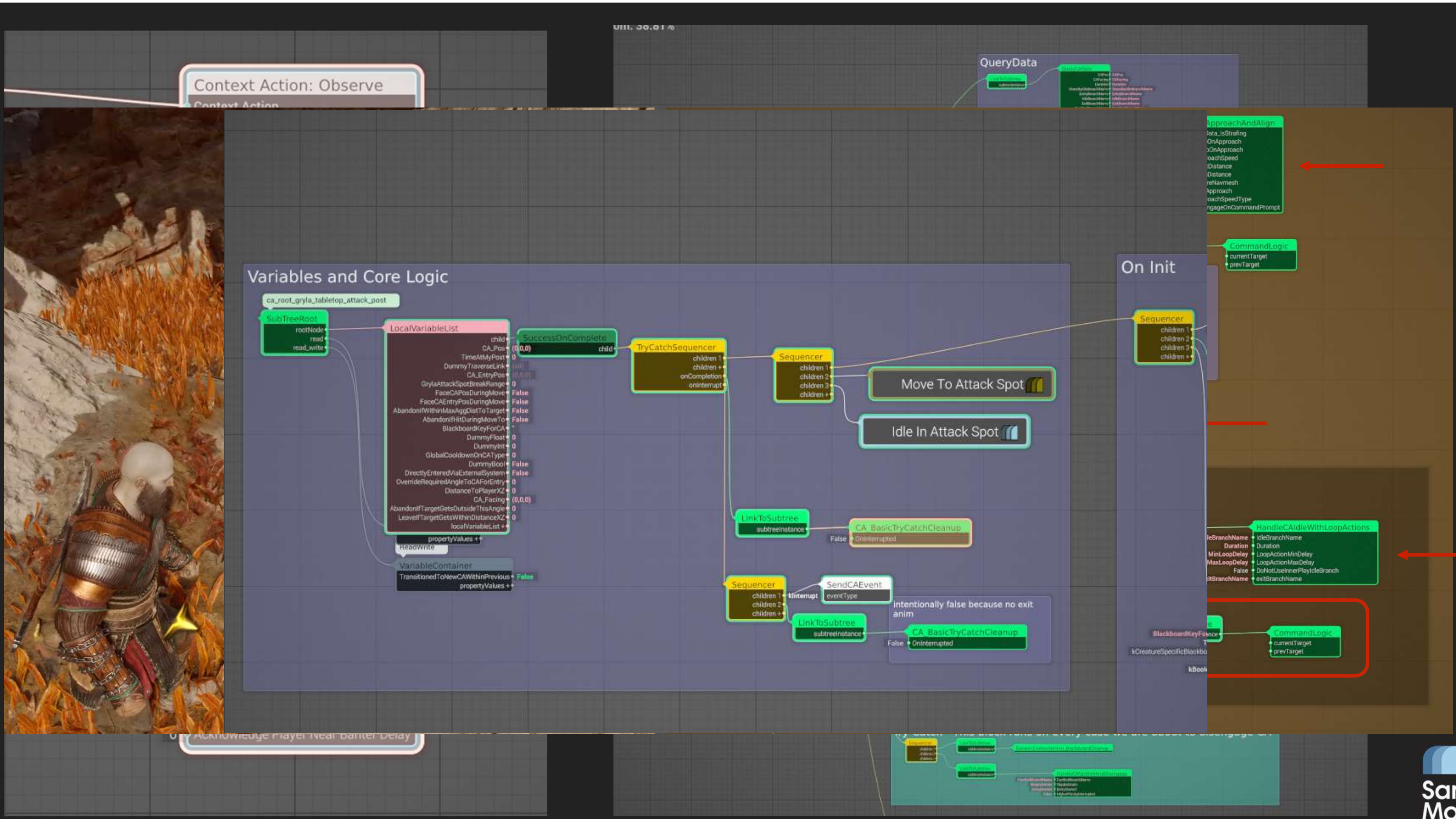
So how does this work on behavior tree side
Lets look at this part of the companion btree for example where you can see the actions AI can do in priority order
If we look at the context action setup closely you will see that we run a search and execute node in parallel. The execute node is the one we have already seen. It has the ability to inject the behavior subtree from the creature's current CA, if they have any. Running search in parallel allows the AI to still dynamically react to new search results
The search is a score based prioritizing things like distance, priority etc. The tech for it was also designed to accommodate different type of searches an AI creature might need.
- Auto engage Search here is intended to realize cases where AI stumbles upon something while naturally navigating through space. This is more like Atreus stopping to pick something up from the ground while following player naturally. Under the hood it is Atreus getting in the range of the placed context action while his current intention might be something like "following player"
- Second search node had more intention behind it. This is more like a Context Action on demand, rather then stumbling upon. Think about a case like a Grim deciding to get on high ground in the natural flow of its combat decision making. So they specifically look for all the ceiling hang context actions around them at that moment.
- there is a third hidden case where an AI is forced to a context action based on a current gameplay beat. And both of these nodes automatically prioritize the forced CA. Like when player presses the interact button to climb up and we want Atreus to immediately follow, we force that follow Context Action so his btree can pick it up right away regardless of search parameters.
So this is how the main brain operates but what happens in the injected behavior tree?



PlayBranch once to be in Idle if not handled by elsewhere, then if CAHasLoopAction just run forever and don't bother re-checking the PlayBranch. Else if no CAHasLoopAction then just succeeds when PlayBranch succeeds

Most common use cases of regular context actions (especially the ones that follow the more traditional approach, enter, idle, exit sequence) use the same main injected subtree. In its most simple form it looked like this :

When we go into what each of these subtree does (entry and idle for example), you can see they run these operations reading the values set by the level scripting. And each one can be expanded more to see more detailed logic of operations like triggering the animation to play, or keeping duration timer and more



Santa Monica Studio

So toggling these VS values as you saw before was one way of customization, but it is somewhat limited. But if you recall we had the ability to assign different subtrees to context actions, so we used that as another customization avenue

If we look at this behavior tree we just saw again, it is all comprised of many subtrees. So this Atreus context action was actually calling into another subtree shared many of the first subtrees, while expanding it for additional shooting behavior

For even more custom setups, we created completely different subtrees. For example, this very specific grayla context action btree that it used while in combat to attack from certain points. It supports a very specific behavior without much of the regular entry/idle/exit flow of regular CAs. This way AI in its core still goes through same streamlined process but can still run a very specific behavior in that instance

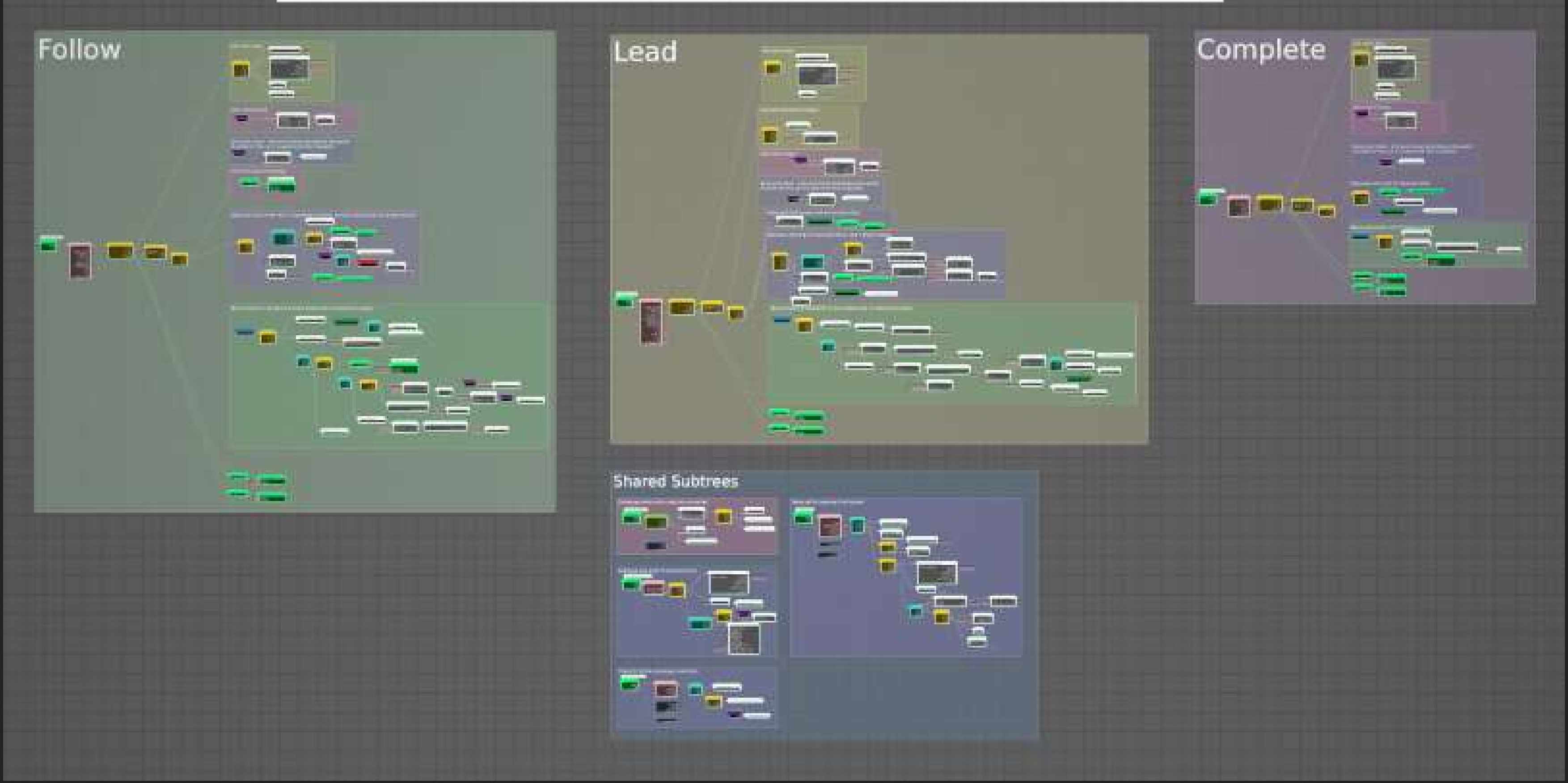
This mixed use of Visual Scripting and Behavior Tree injecting gave us the ground to support the two user in mind :

1. Level designers easily dropping Context Action modules and toggling only the options they need for that instance and

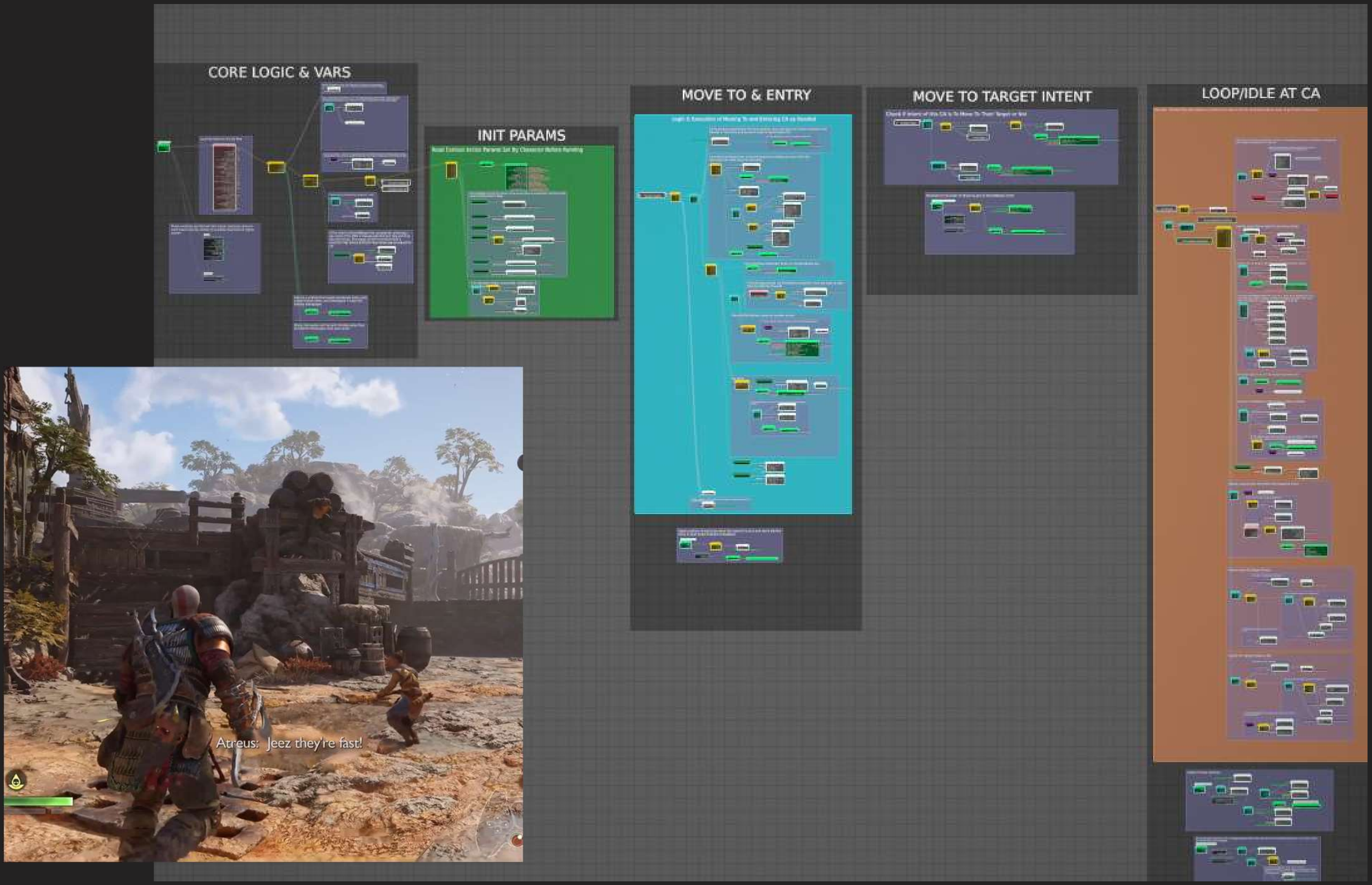
2. technical designers who can easily setup custom behaviors and mix/match modules to create new ones for all design use. All that without programming support

And looking at these subtrees it might seem simple and straightforward but these subtrees can get ...

SALAAR KOHARI ♦ PROGRAMMER
Companion Traversal in 'God of War Ragnarök' ♦ Programming
FRIDAY, MARCH 24 ♦ 10:00AM – 11:00AM ♦ ROOM 2002, WEST HALL

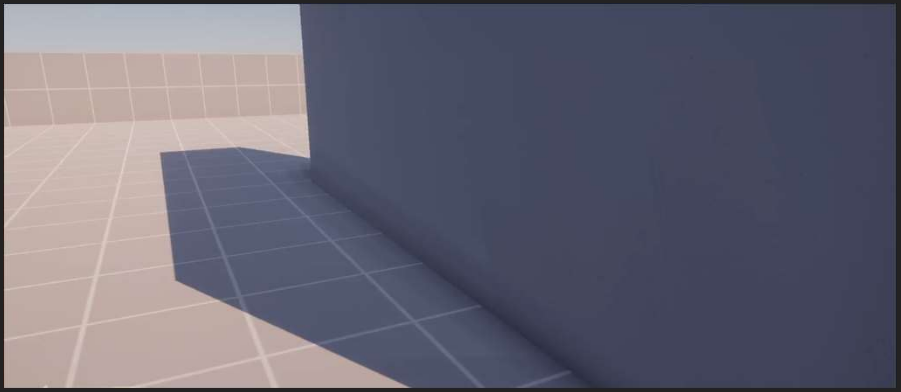


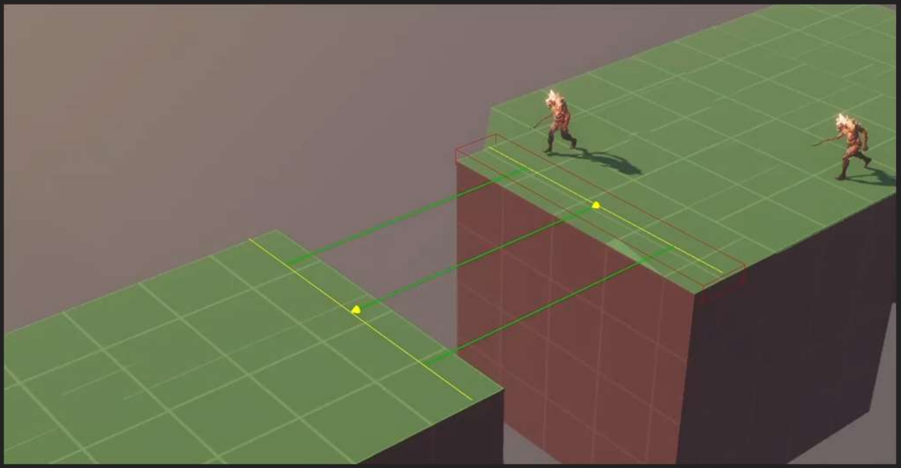
...VERY complicated, very quickly.
Here is a zoomed out version of the traversal behavior context actions for AI. Which I will not dive much into but if you are interested go check out Salaar's talk



Here is another example. This is a zoomed out look of the subtree that can handle enemy behavior on context actions. Remember those Grims who would relentlessly spit at you from atop the walls, you can thank this behavior tree for that.
Grims were actually and interesting case and introduced a lot more complication than this specific btree

Enhanced Movement and Context Actions





- Support entry/exit animations from multiple angles and heights
  - Ability to adjust the placement of these entry positions
  - Ability to adjust the size and shape of valid entry/exit zones
- Support direct transitions from one Context action to another

Santa Monica Studio

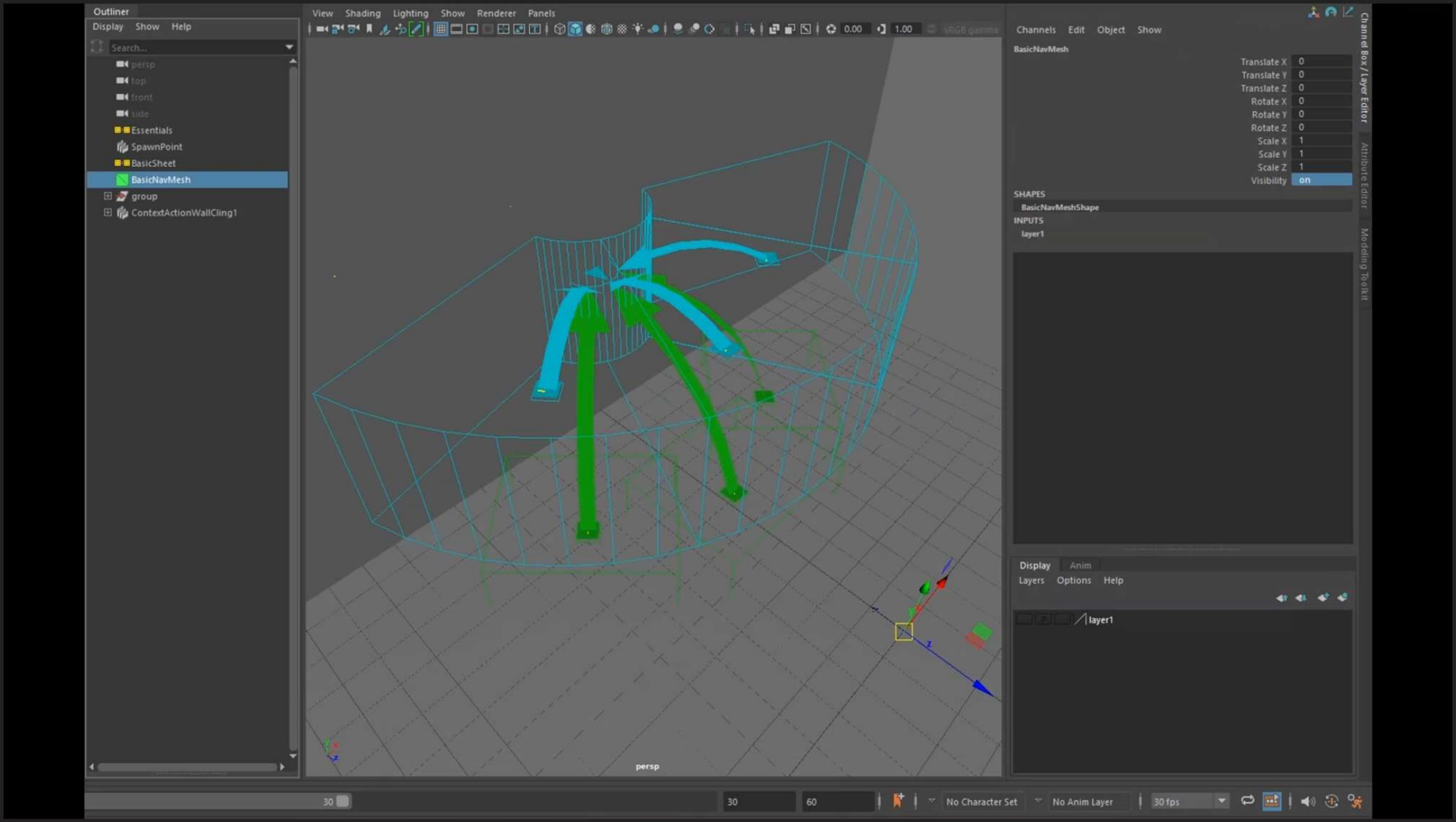
Fundemantally, they follow the same thing – approach, entry, idle, exit. But they needed to be doing a little bit more than jut playing an entry anim, they needed to jump to an offnavmesh location and align with the wall and similarly exit to a safe spot on the ground. More importantly :

- Support entry from multiple angles and heights to accommodate for our high animation fidelity requirements
- Also give users the ability to mark safe entry/exit zones, adjust placement, shape and size of these zones within our metrics
- For future design plans, we also needed to support a way to go from one wall hang to another without having to come down and go back up

So we looked at our existing systems and decided to improve upon an already existing module that was doing kind of a similar thing in terms of creating connections through non-navmesh area – traverselinks : this is what they looked like in maya

They were representing a text defined set of data that looked like this in runtime, two ends that connects two edges of navmesh, an animation associated with this for AI to play, and an area attached to both ends that mark valid entry and exit zones

And not so surprisingly, this simple system was a beast of its own by the time we were done with it



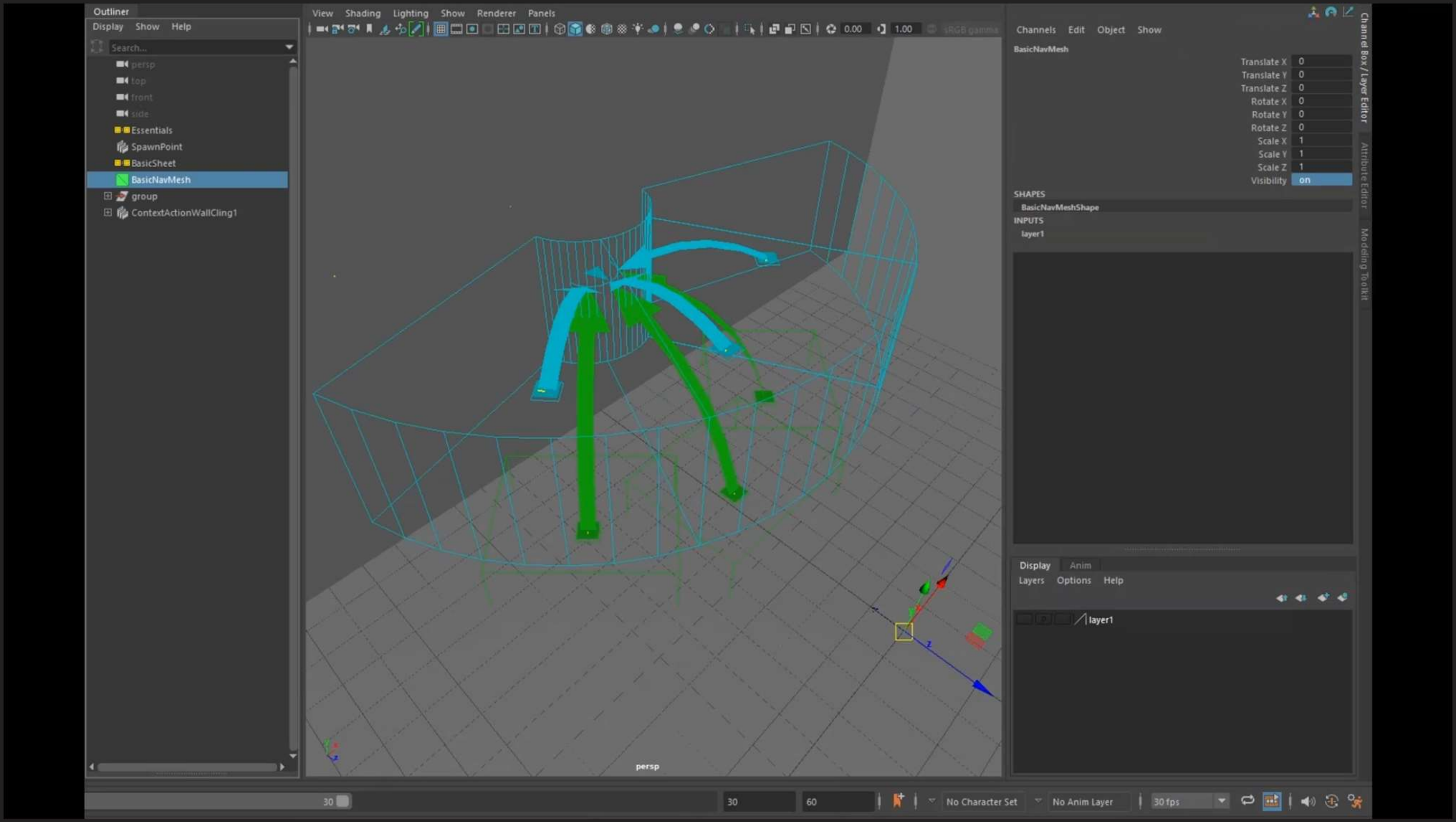
BasicNavMesh

Translate X	0
Translate Y	0
Translate Z	0
Rotate X	0
Rotate Y	0
Rotate Z	0
Scale X	1
Scale Y	1
Scale Z	1
Visibility	on

SHAPES
BasicNavMeshShape
INPUTS
layer1

30 60

So the end result was context actions with a bunch of traverselinks that are connected to them. And we created templates out of them (kind of like prefabs as a more common term) to be placed by designers.
Tech designers would be creating each of these templates:
- by starting with a context action – regular setup of assign category and a subtree to run.
- then they would place a traverselink and chose the type This is mostly a list of animation sets to play for both forward and backward.
- Then traverselink is connected to Context action which automatically moves them as child in hierarchy
- As you can see we usually ended up more than one link, to represent different animation coverage angles, so these metrics we pre discussed metrics with animation team
- Each traverselink came with its valid zone, that can be different type as you can see based on the need.
- Once the position of each traverselink is also adjusted as needed (yellow arrow represent the angle to be facing at the end of anim), the type is selected and a template can be created

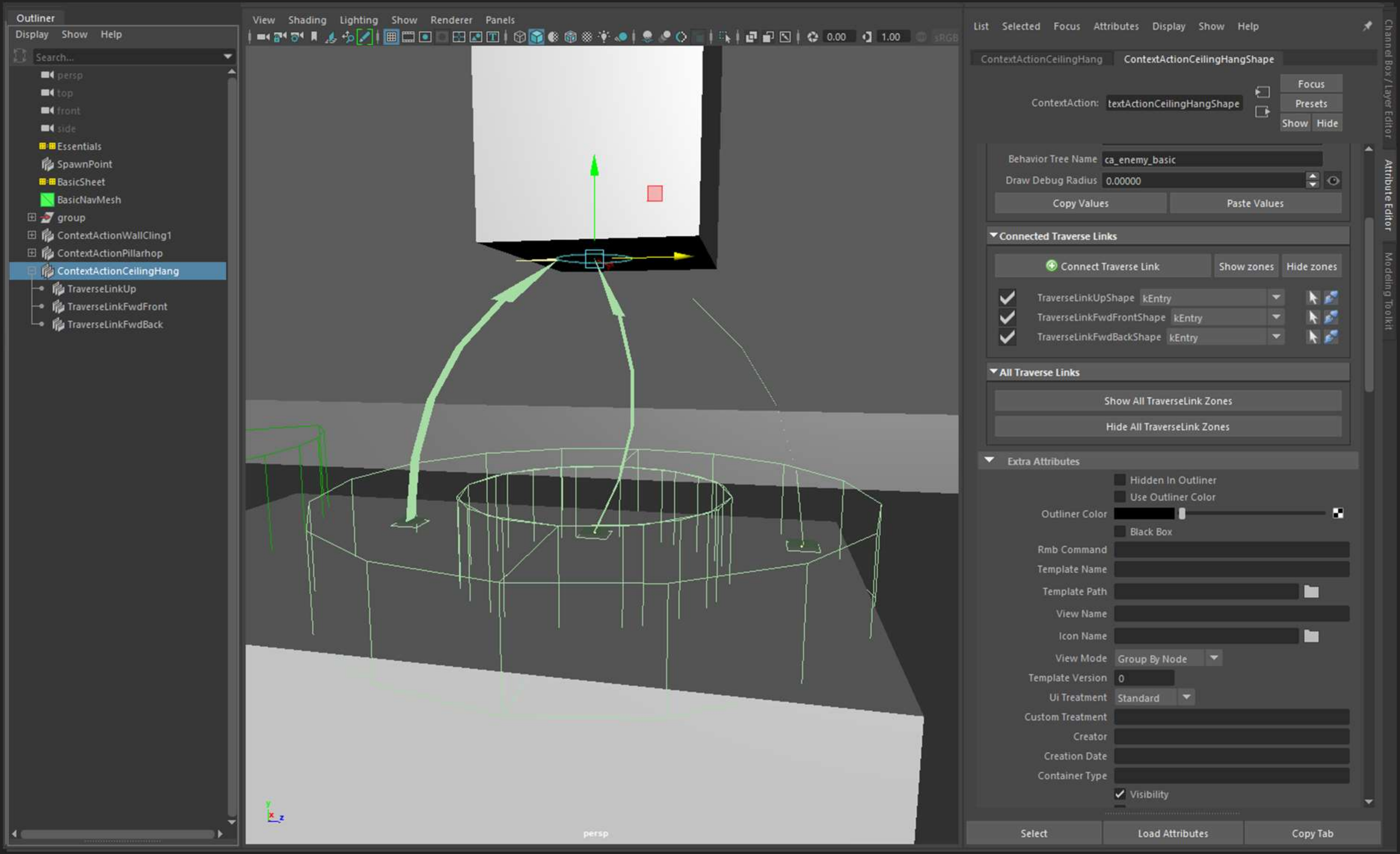


Translate X 0
Translate Y 0
Translate Z 0
Rotate X 0
Rotate Y 0
Rotate Z 0
Scale X 1
Scale Y 1
Scale Z 1
Visibility on

We supported transition requirement in a similar way by simply providing an interface to assign a type to traverselinks.

Templates had ability to be updated

Once ready, designers can just place them in the environment and manipulate that instance as they see fit by disabling certain links, or manipulating values like the position, angle and size of the zones



ContextAction: textActionCeilingHangShape

Behavior Tree Name: ca_enemy_basic

Draw Debug Radius: 0.00000

☑ TraverseLinkUpShape kEntry

☑ TraverseLinkFwdFrontShape kEntry

☑ TraverseLinkFwdBackShape kEntry

□ Hidden In Outliner

□ Use Outliner Color

Outliner Color

□ Black Box

Rmb Command

Template Name

Template Path

View Name

Icon Name

View Mode: Group By Node

Template Version: 0

Ui Treatment: Standard

Custom Treatment

Creator

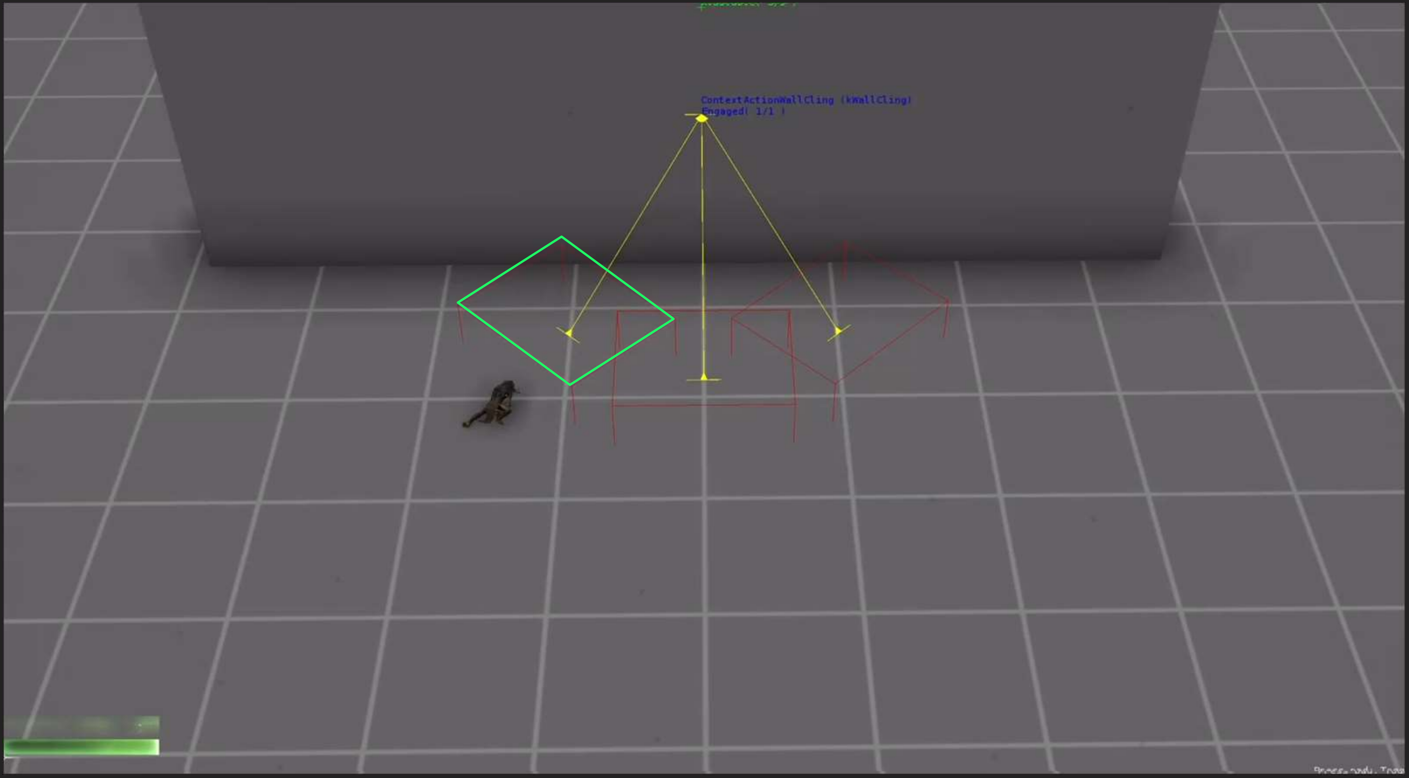
Creation Date

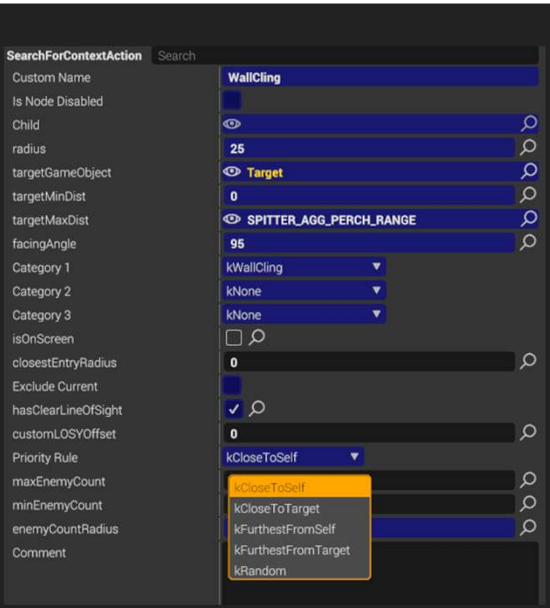
Container Type

☑ Visibility

Here you can see the example of our "pillar hop" template that comes with full set of animation support. Design then is expected to pick and chose based on the environment and adjust as needed. And another template for ceiling hang that is slightly simpler.

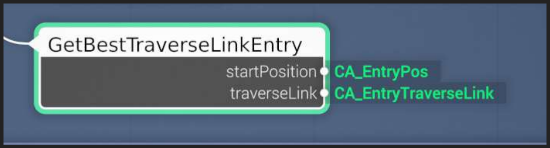
Templates needed updates as we go through the production, so we also needed to support the updates of existing instances in the most design friendly way possible





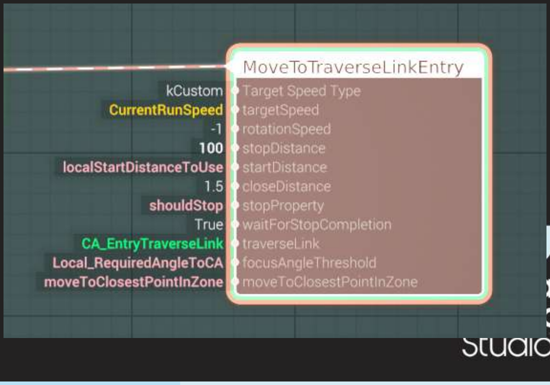
SearchForContextAction Search

Custom Name	WallCling
Is Node Disabled	
Child	
radius	25
targetGameObject	Target
targetMinDist	0
targetMaxDist	SPITTER_AGG_PERCH_RANGE
facingAngle	95
Category 1	kWallCling
Category 2	kNone
Category 3	kNone
isOnScreen	□
closestEntryRadius	0
Exclude Current	
hasClearLineOfSight	✓
customLOSYOffset	0
Priority Rule	kCloseToSelf
maxEnemyCount	
minEnemyCount	
enemyCountRadius	
Comment	



GetBestTraverseLinkEntry

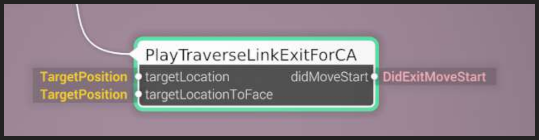
startPosition	CA_EntryPos
traverseLink	CA_EntryTraverseLink

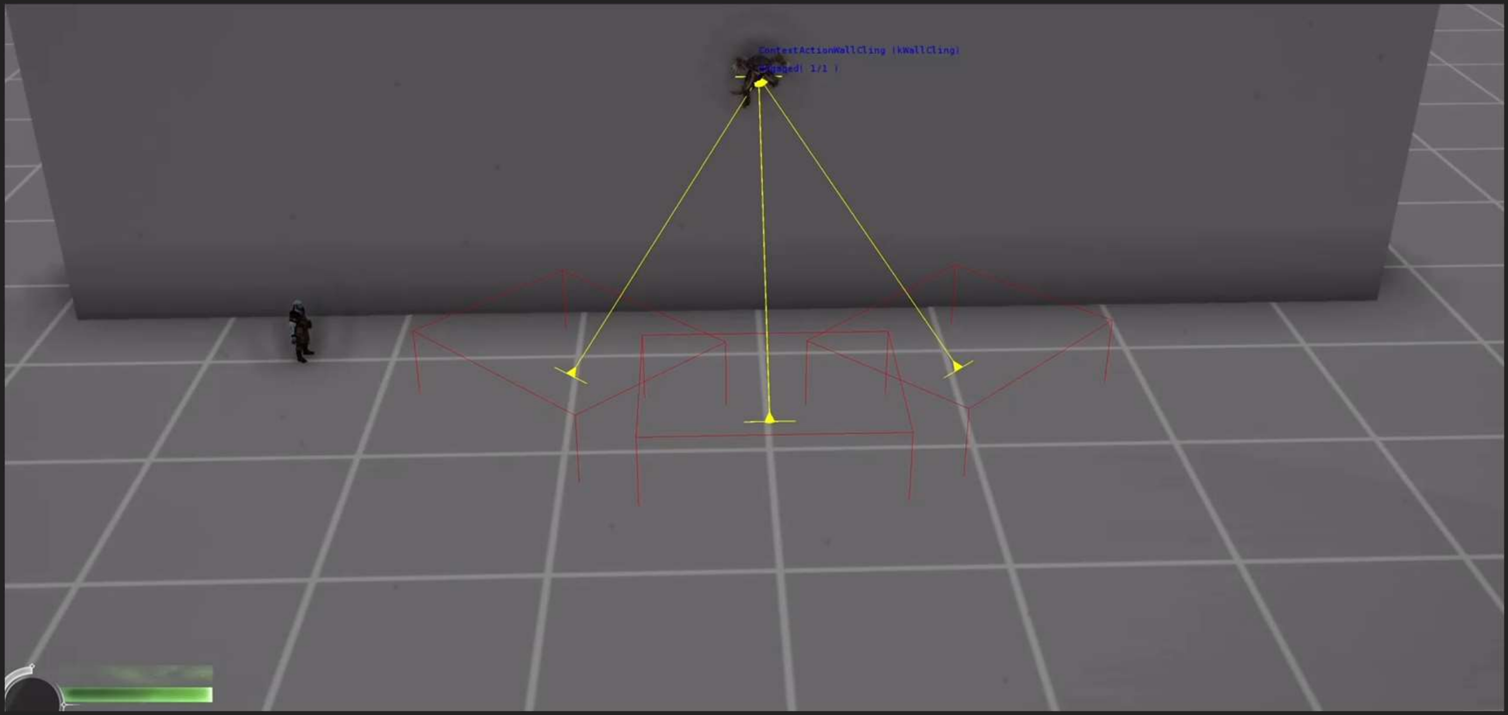


MoveToTraverseLinkEntry

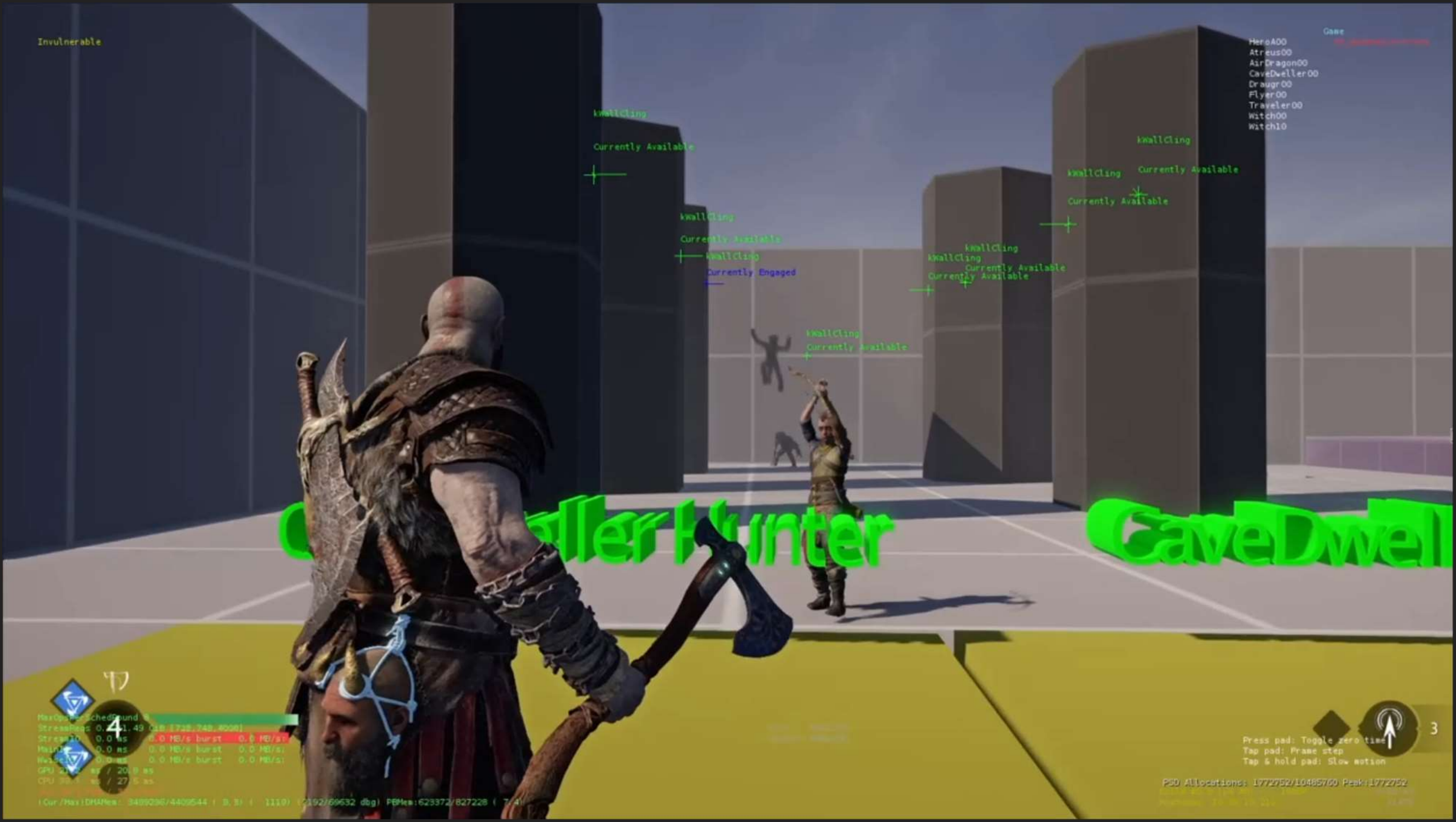
kCustom	Target Speed Type
CurrentRunSpeed	targetSpeed
-1	rotationSpeed
100	stopDistance
localStartDistanceToUse	startDistance
1.5	closeDistance
shouldStop	stopProperty
True	waitForStopCompletion
CA_EntryTraverseLink	traverseLink
Local_RequiredAngleToCA	focusAngleThreshold
moveToClosestPointInZone	moveToClosestPointInZone

So if this is what you see in Maya, how does all this playout in runtime?
As mentioned before, it starts with an intentional search of a specific type of context action.You see a bunch of different options here but We did not start with this many options as you can guess. As we playtested enemy behavior, we started adding more to prevent them from doing silly things like walking behind the player to enter into a wall cling that is in front of him.
When CA is found there is a bit of an extra step this time. If it is a CA off navmesh like this one, we need to choose the best entry link based on distance (NEXT)
Then AI needs to move to the entry zone, which had its own specific node with some options and once this node succeeds, all that is left is to play the anim. EXCEPT we cannot just play because AI can end up at a completely different position based on his pos and angle so we warp them to position identified by Context action (that yellow arrow)as we do so
The rest is the good old idle behavior (+ their combat attacking behavior running in parallel with awareness that creature is in a CA)

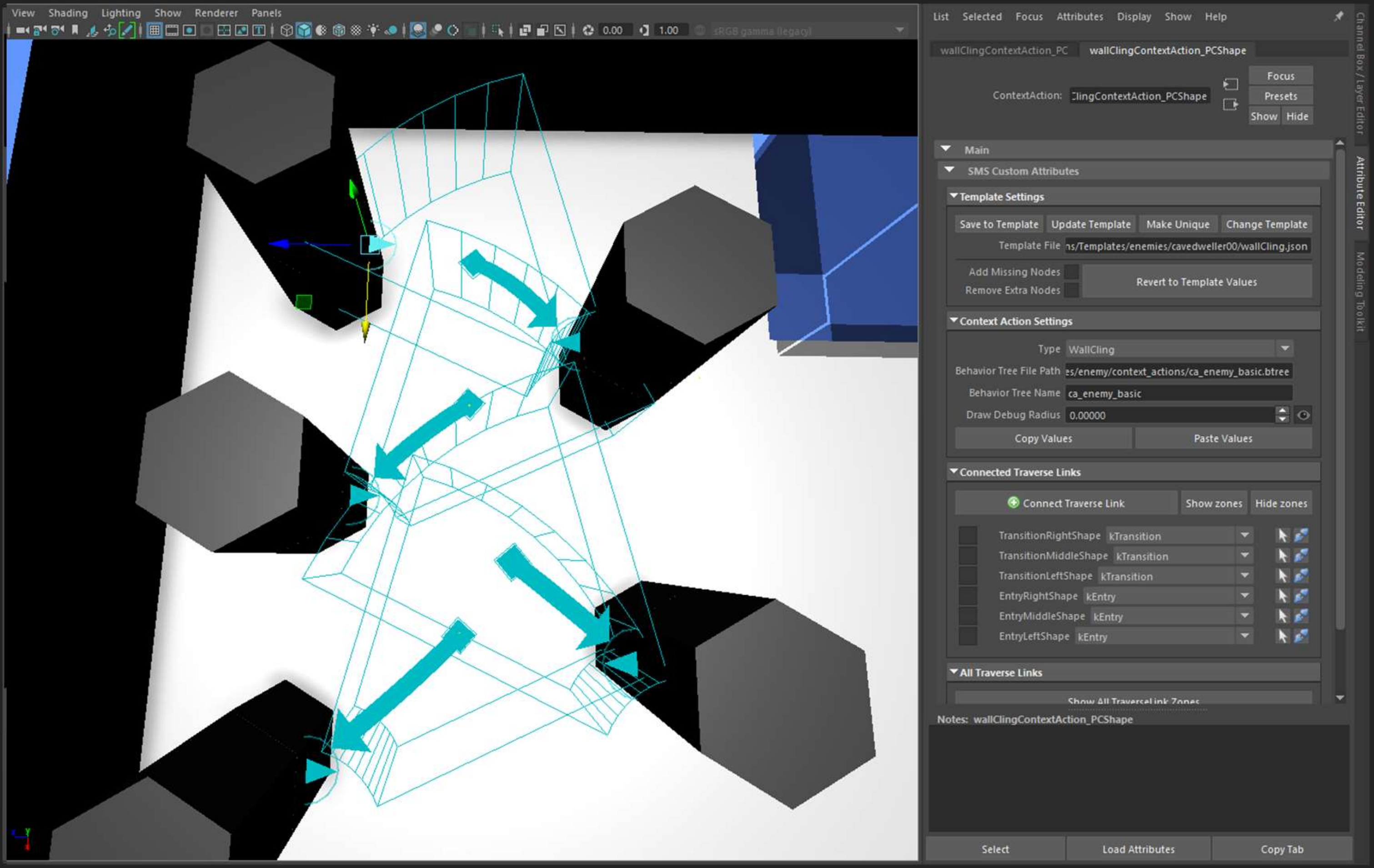




Exit also had similar set of rules. These zones were already deemed safe by design, so they could just use any to exit safely back on navmesh by using this node. Here target location and facing were optional parameters. If nothing is passed we just pick a random link.
But lets say this Grim is targeting Brok here. Then it would make sense for AI to get as close to their target as possible. So we not only chose the closes zone, we also warp them (NEXT) to the closest position within the zone to their target. Similarly, we had the ability to make them face to their target if requested to achieve more believable AI in combat



So lets get back to this one requirement from before, supporting direct transitions between context actions. I even showed you how to set those up, what were they for? They were actually intended to support some cool movement in the new combat arenas. These context actions were not only used as "perch and attack" but also as means of transport as they followed their target around



ContextAction: ClingContextAction_PCShape

Template File: ns/Templates/enemies/cavedweller00/wallCling.json

Add Missing Nodes □

Remove Extra Nodes □

Type: WallCling

Behavior Tree File Path: es/enemy/context_actions/ca_enemy_basic.btree

Behavior Tree Name: ca_enemy_basic

Draw Debug Radius: 0.00000

□	TransitionRightShape	kTransition
□	TransitionMiddleShape	kTransition
□	TransitionLeftShape	kTransition
□	EntryRightShape	kEntry
□	EntryMiddleShape	kEntry
□	EntryLeftShape	kEntry

Notes: wallClingContextAction_PCShape

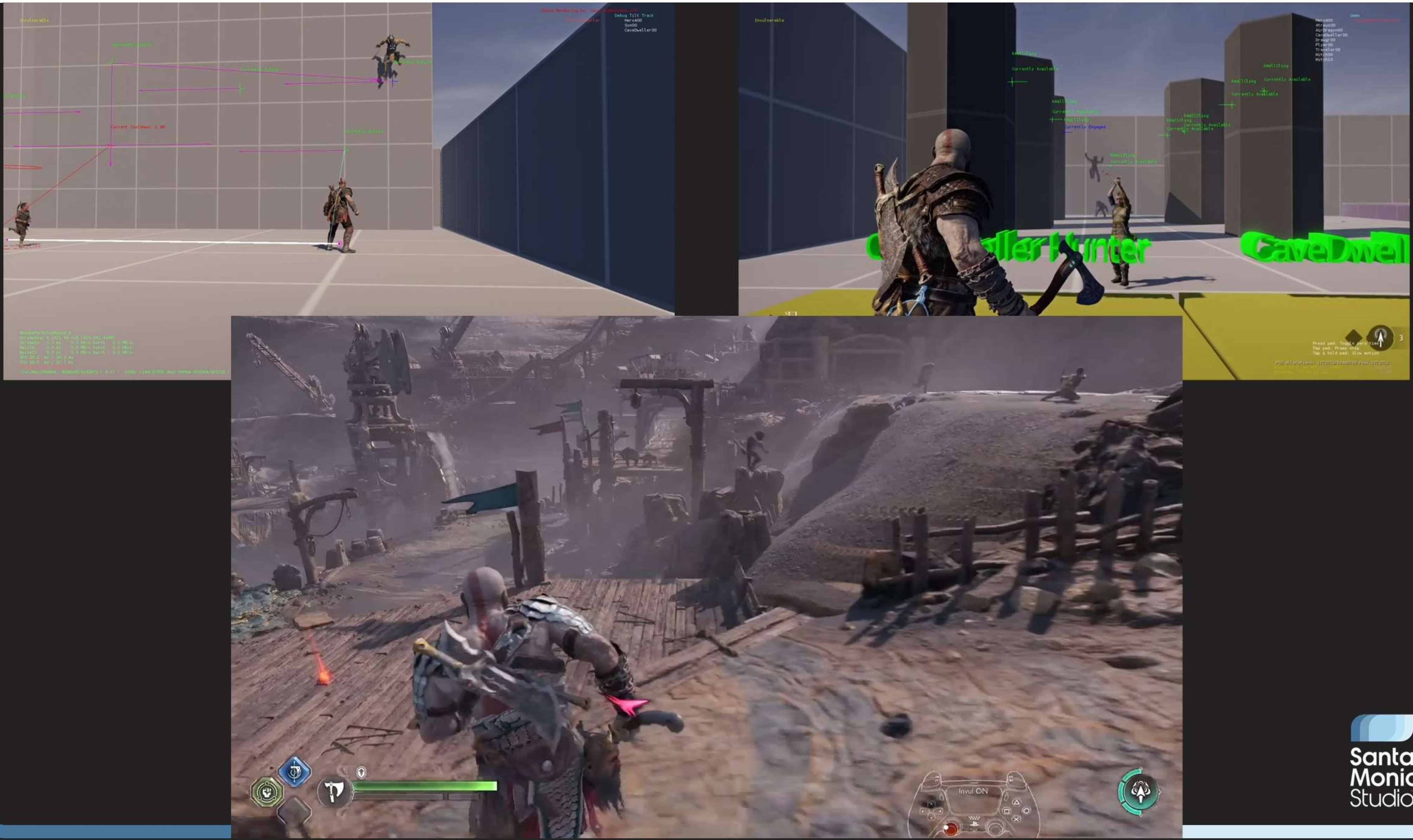
And an example setup to achieve this would have looked like this

If you have 2 CAs setup like this in 2 pillars and their entry moves setup as seen.

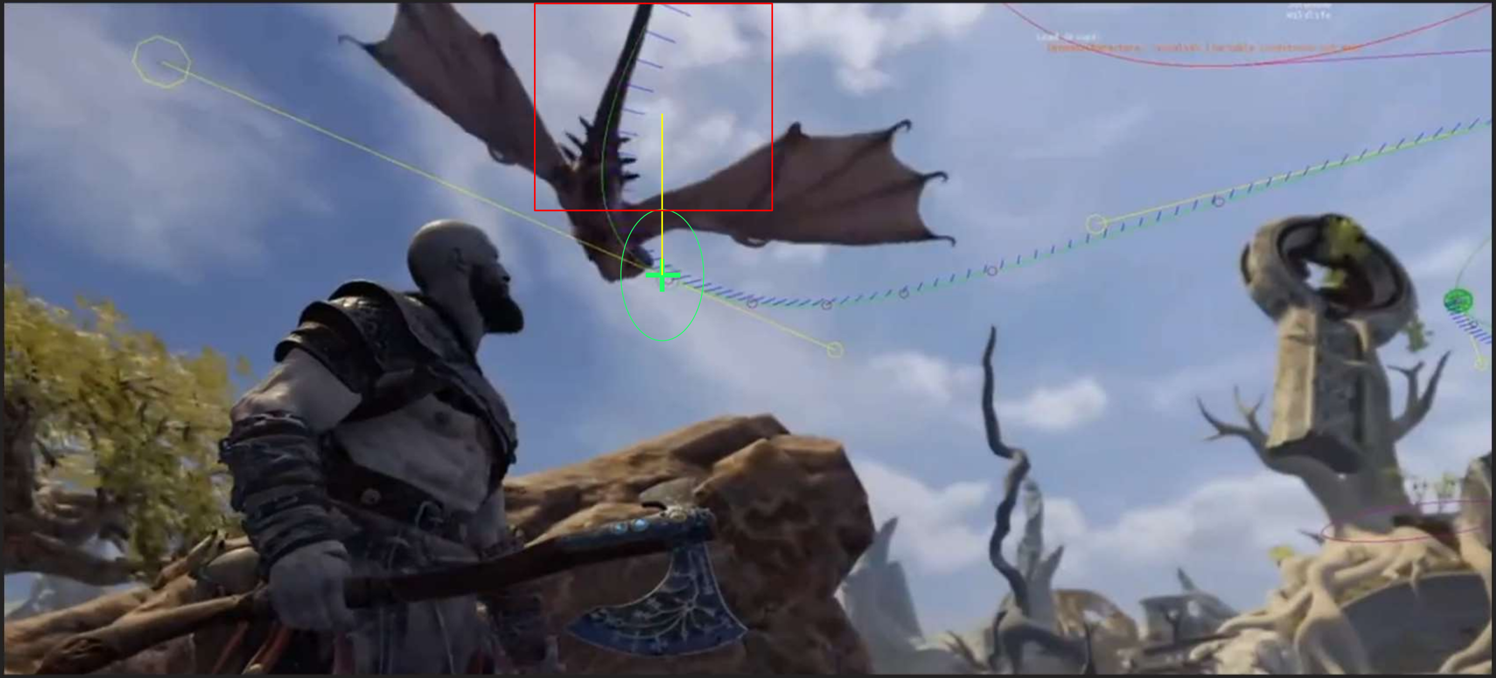
If you also turn on transitions from the template, and lets hid entries for now for clarity They actually represent areas that a creature can transition TO the CA they are associated with

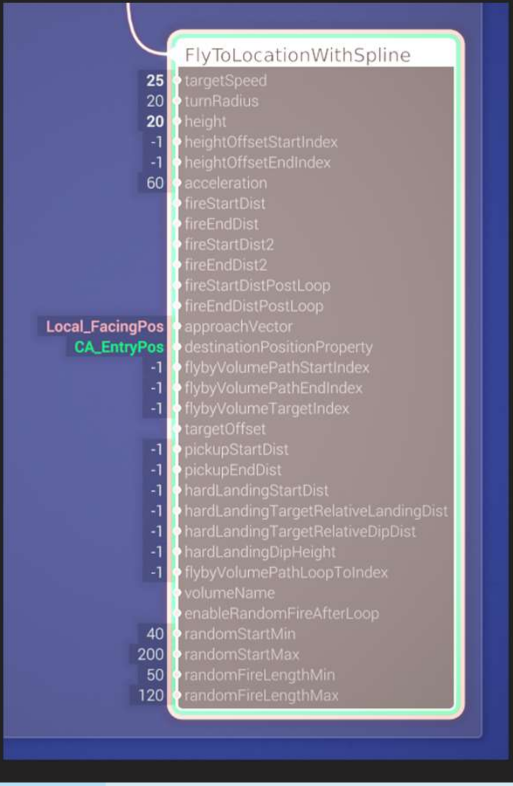
SO Looking at just these 2 CAs and 3 transition animation possibilities, designer would need to turn one on that can encapsulate as seen from this top down view

If we do the same thing for the other 3 context actions you have seen in that video, you achieve what you just saw in that video



Grims were actually one of the earliest creature prototypes we worked on. Here you can see a very early movement test I put together without any of the tools that I had just talked about, back when they did not even have their own model. And this is the video you have seen, something we used to prove out that our new tools and systems were able to accommodate for a setup like this and finally an example encounter arena using this setup in shipped game

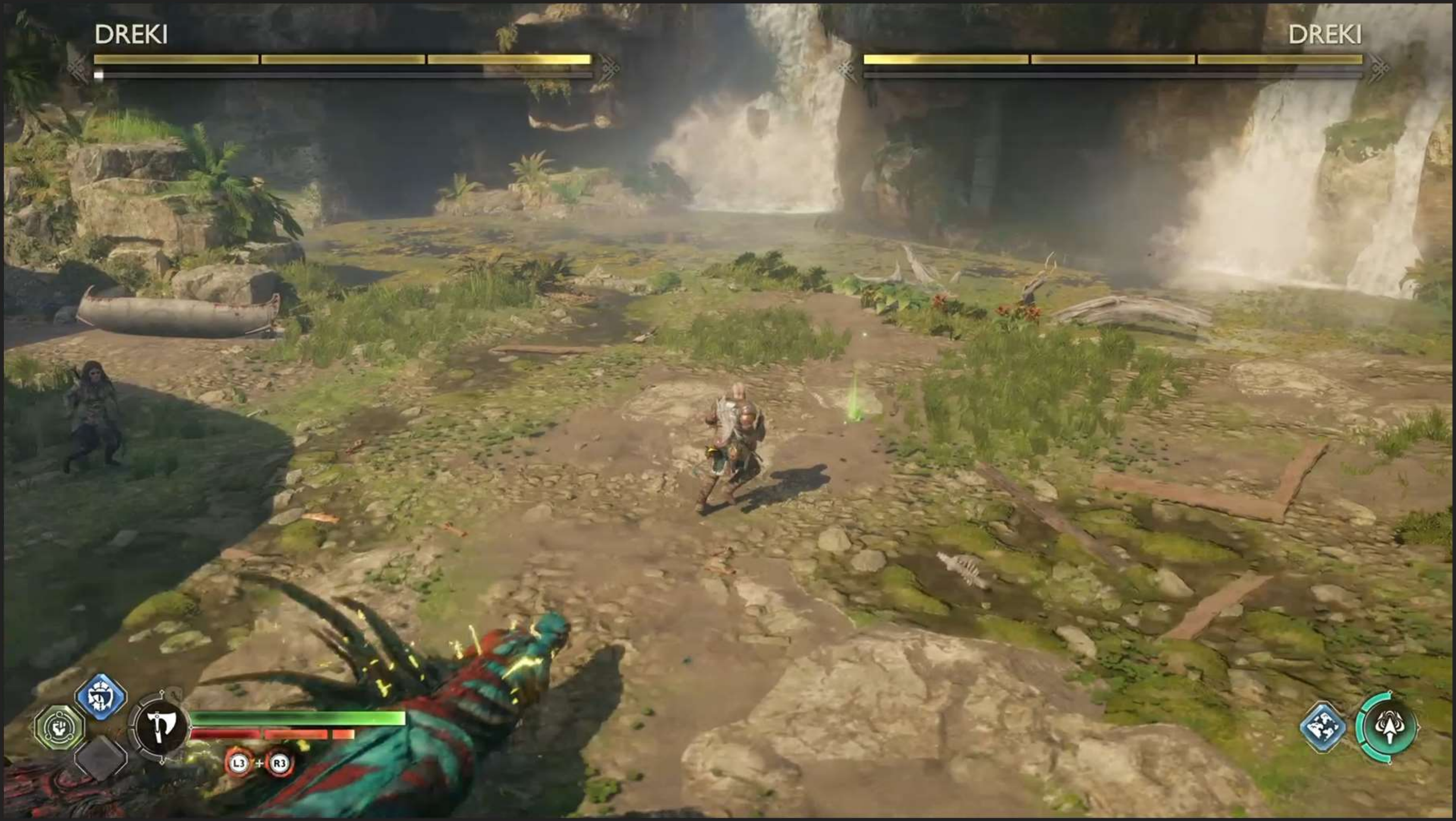




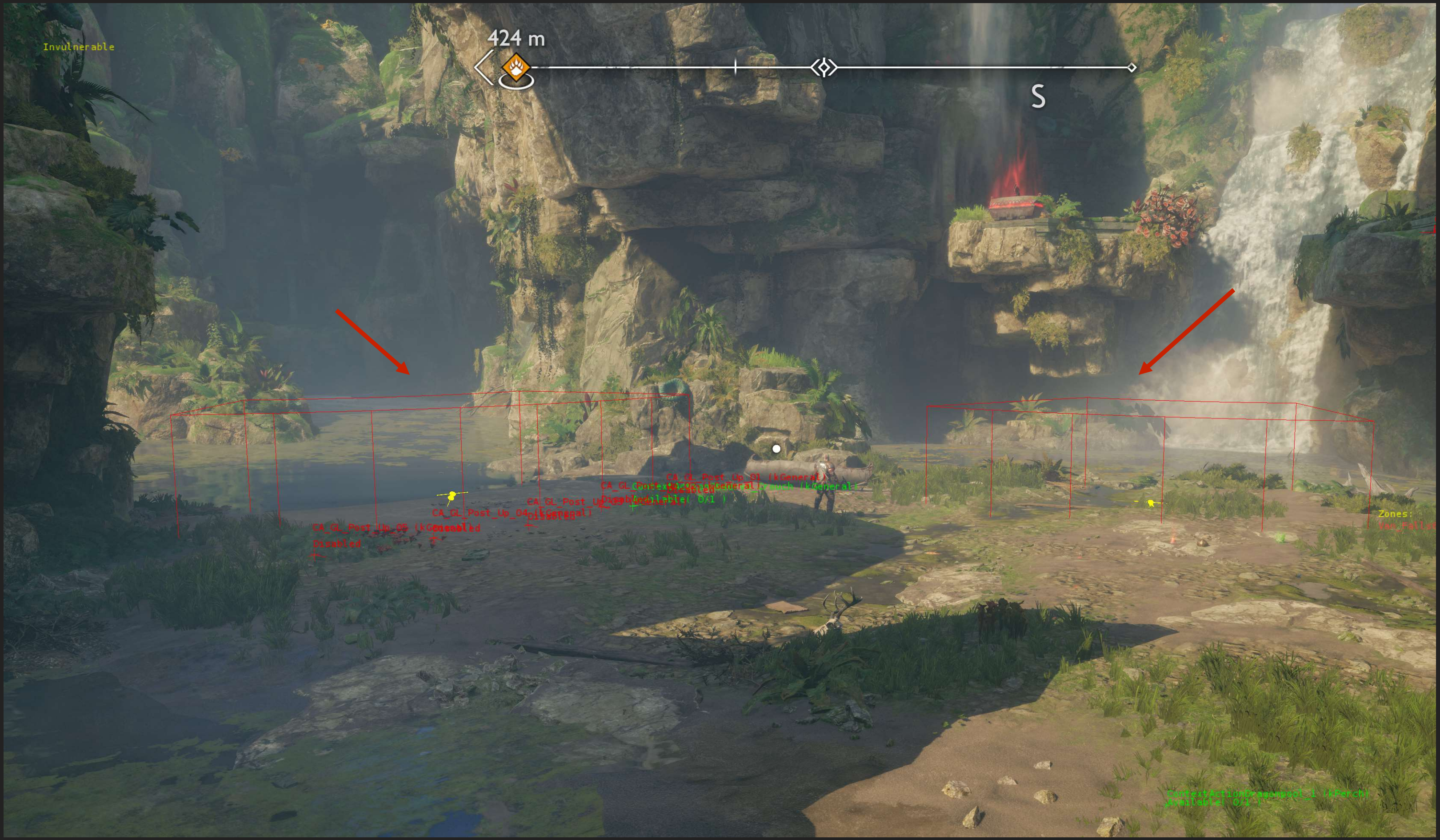
FlyToLocationWithSpline

25	targetSpeed
20	turnRadius
20	height
-1	heightOffsetStartIndex
-1	heightOffsetEndIndex
60	acceleration
	fireStartDist
	fireEndDist
	fireStartDist2
	fireEndDist2
	fireStartDistPostLoop
	fireEndDistPostLoop
Local_FacingPos	approachVector
CA_EntryPos	destinationPositionProperty
-1	flybyVolumePathStartIndex
-1	flybyVolumePathEndIndex
-1	flybyVolumeTargetIndex
	targetOffset
-1	pickupStartDist
-1	pickupEndDist
-1	hardLandingStartDist
-1	hardLandingTargetRelativeLandingDist
-1	hardLandingTargetRelativeDipDist
-1	hardLandingDipHeight
-1	flybyVolumePathLoopToIndex
	volumeName
	enableRandomFireAfterLoop
40	randomStartMin
200	randomStartMax
50	randomFireLengthMin
120	randomFireLengthMax

Air dragons also used Context Actions because they shared a similar problem of marking safe perch points to land to from their flight.
If you look at this flying dragon in this video, he is currently engaged with perch context Action here (Next)
His behavior tree for this context action is setup to use a special node (instead of a regular walk on navmesh) that was lovingly created by Geoff Harrower using some dynamic spline system. You can see the debug draw for it in here.
As he continues flying, right at this point he actually enters the entry zone for traverselink, same way the Grims would do and the plays its entry move

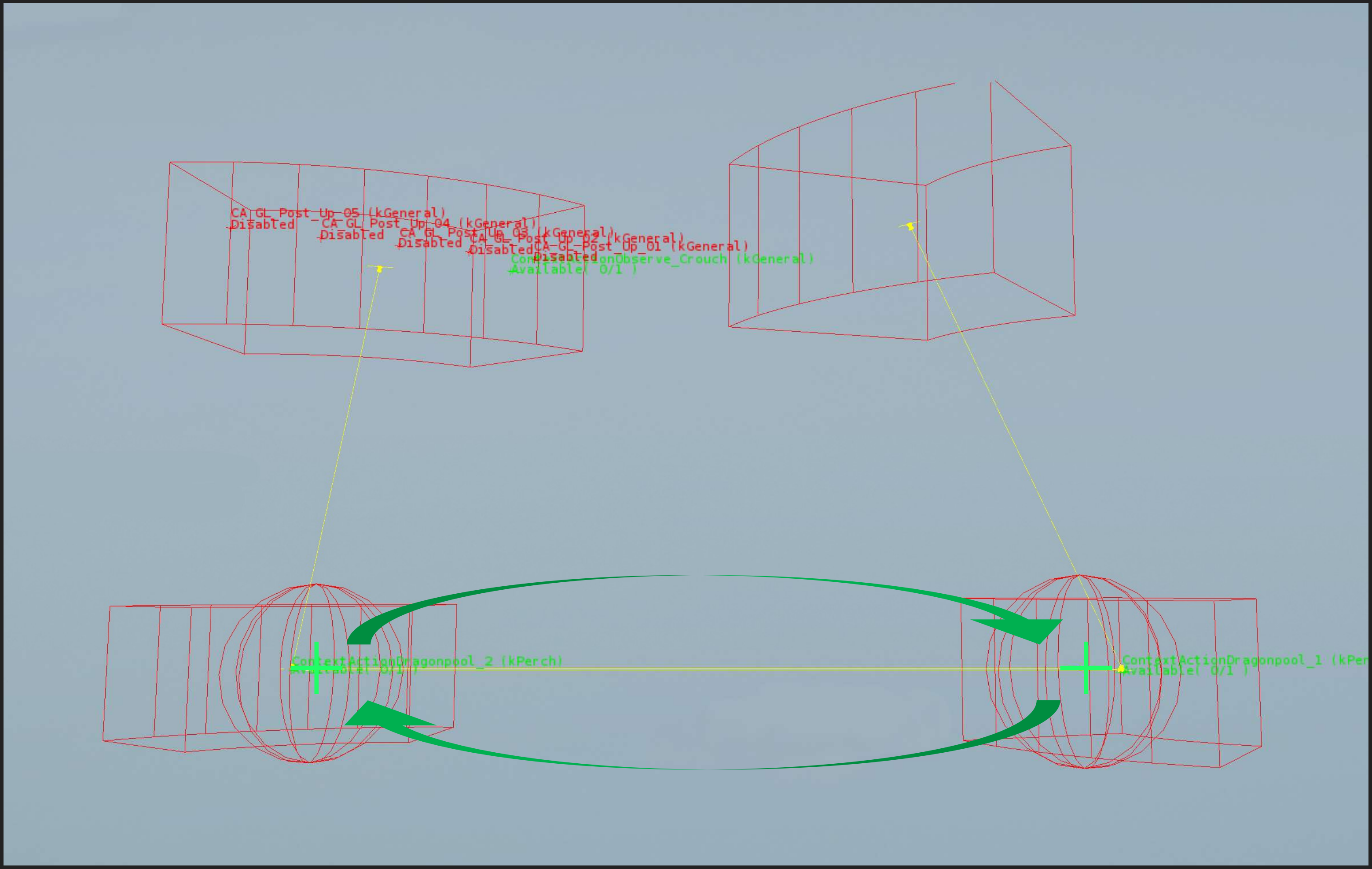


Drekis also used context actions for their in water behavior as you see in this video


Invulnerable
424 m
S
CA_GL_Post_Up_01 (kGeneral)
CA_GL_Post_Up_02 (kGeneral)
Disabled
CA_GL_Post_Up_03 (kGeneral)
Disabled
CA_GL_Post_Up_04 (kGeneral)
Disabled
CA_GL_Post_Up_05 (kGeneral)
Disabled
Available (0/1)
Zones:

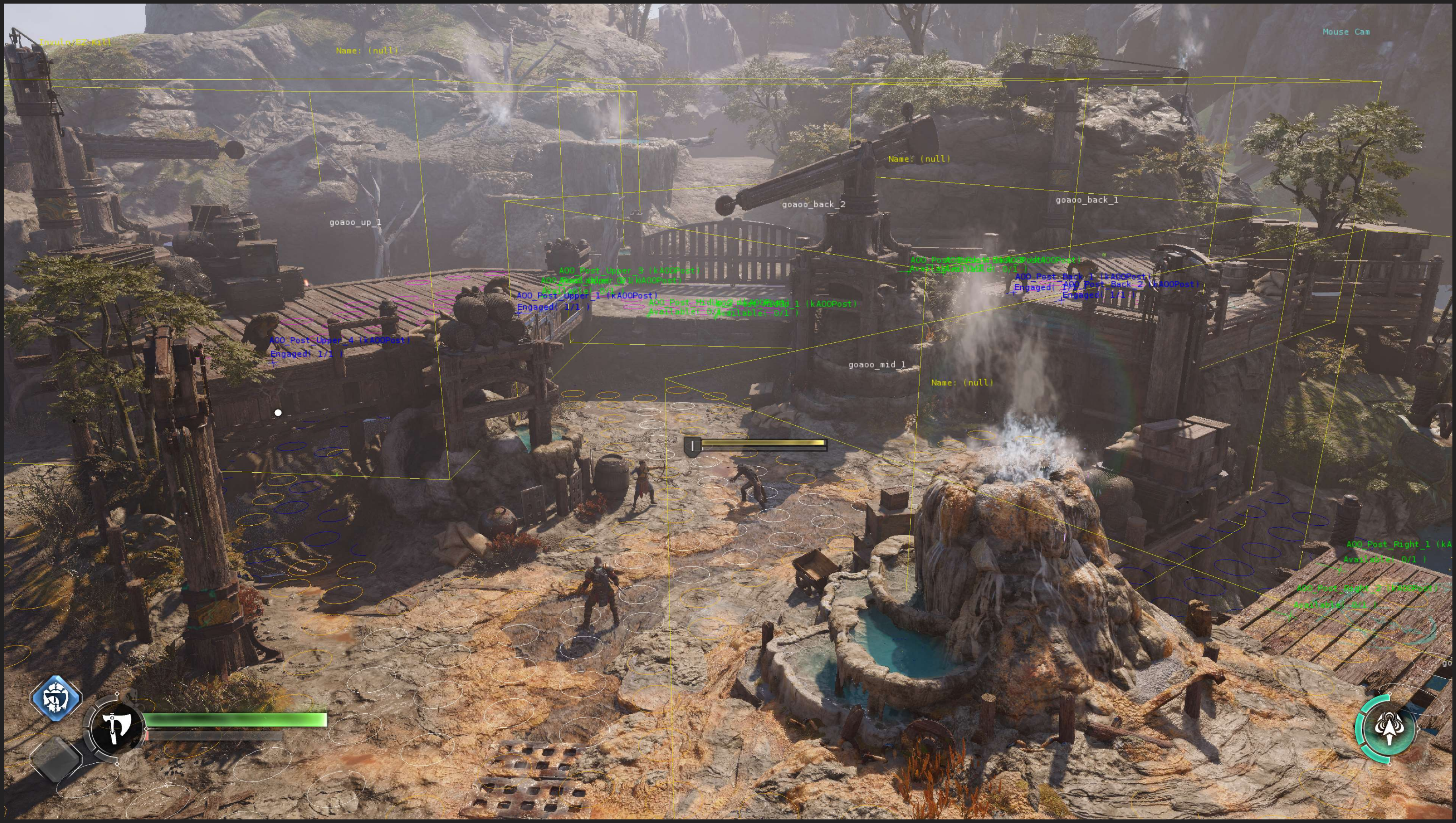
These behaviors were supported by Context Actions in these "pools" as we called them. If you look at this image you can see the (NEXT) these two entry zones that go into two pool context actions in water.

A behavior requirement for Drekis was for them to be able to dive into one pool and resurface on another one, which as you can guess called for transition traverselinks. So if we were to go underground from this look ...


CA GL Post Up 05 (kGeneral)
Disabled
CA GL Post Up 04 (kGeneral)
Disabled
CA GL Post Up 03 (kGeneral)
Disabled
CA GL Post Up 02 (kGeneral)
Disabled
CA GL Post Up 01 (kGeneral)
Disabled
ContextActionObserve_Crouch (kGeneral)
Available(0/1)
ContextActionDragonpool_2 (kPerch)
Available(0/1)
ContextActionDragonpool_1 (kPer
Available(0/1)

This is what it looks like in game. Tow context actions can be seen here, with two transition links going to them encapsulating each other
So if you think about this, we actually did not need them animating from one CA to another, they were invisible to the player. So what we needed was essentially them warping from one Context Action to another. But instead of introducing a custom setup, we chose to use traverselinks with an animation that was only 2 frames of static poses. . Since transition time is completely driven by animation, they would essentially warp from one to another this way

Callout the transition traverse link and how we achieved instantaneous warp



Santa Monica Studio

Before I end this, I also quickly want to touch on one of the problems I brought up earlier regarding the transition from GOW2018 to Ragnarok -> the flatness of our positioning systems, when our plan was to have combat arenas with many platforms like this one

Well, we actually chose not to solve this problem for this project. Updating our positioning system fully to account for the verticality in the way we would have wanted was too much to take on top of the major works I had mentioned today.

HOWEVER we did come up with a simpler solution that would do the job for the planned encounters of GOWR

First We did some minor improvements for positioning to be able to recognize different layers (show image with just fight knowledge) – more sensitive to height and pink ones

Then Created this simple system called "Area of operations" (show image) – It was essentially boxes encounter designers would place and assign AI to. Once an AI is assigned to one, AI's navmesh would be cut off to the limits of these boxes – This helped minimize them going up and down if they are not actively attacking, also encourage player to move up and down as well

We also had Context Actions for their posting behavior in this zone and adjusted Context Action search to only give results for CAs that are within AOOs if assigned

This combined with the enhanced movement context actions and traversals gave you the end result you saw in these fights

So the important thing I would like to point out here was that this was us being realistic with the time we have, without sacrificing content. So I also urge you to think about putting your resources and time to get the bigger gains and Looking for simpler ways out for others to reach desired end result

Needs and requirements :

- Dynamic & vertical combat arenas
- Enhanced movement enemies & more enemy variety
- More companions
- Spaces with more AI agency
- A more

Challenges :

- Positioning
- Systemic world annotation
- Non expandable companion workflows
- Lack of centralized AI decision making
- Lua scripting
  - Performance
  - Stability
  - Validation & Debugging
  - Reading, authoring & maintaining

Solutions :

- Behavior Trees
  - Moving from Lua to Behavior Trees
  - Tools support
  - Key usages of injected behaviors
- Context Actions
  - Achieving modularity & customization
  - Different use cases
- Enhanced Movement and Verticality

We covered a breadth of topics today relating to AI and how we prepared them for Ragnarok to realize the vision
This was a huge team effort across different engineering teams and our partners in design, production, QA teams. Huge shoutout to everyone involved for their support.
And to everyone who gave their feedback to this talk for me to represent our joint efforts best way possible

Thank you!

Göksu Uğur

goksu_ugur

Santa
Monica
Studio

And finally thank you for coming to listen

JOIN US AT GDC 2023

GOD OF WAR RAGNARÖK

BUILD YOUR GOD OF WAR GDC AT:
SCHEDULE.GDCONF.COM

BRUNO VELAZQUEZ ♦ ANIMATION DIRECTOR
DAVID GIBSON ♦ ANIMATION DIRECTOR
ERICA PINTO ♦ LEAD NARRATIVE ANIMATOR
MEHDI YSSEF ♦ LEAD GAMEPLAY ANIMATOR
Animation in 'God of War Ragnarök' ♦ Animation Summit
MONDAY, MARCH 20 ♦ 9:30AM – 10:30AM ♦ ROOM 303, SOUTH HALL

SUE PACETE ♦ SR USER RESEARCHER
Playtesting God of War Ragnarök Accessibility Options ♦ UX Summit
MONDAY, MARCH 20 ♦ 1:20PM – 1:50PM ♦ ROOM 2001, WEST HALL

PAOLO SURRICCHIO ♦ SR STAFF PROGRAMMER
Reinventing the Wheel for Snow Rendering♦ Advanced Graphics Summit
MONDAY, MARCH 20 ♦ 1:20PM – 2:20PM ♦ ROOM 301, SOUTH HALL

BEN HINES ♦ SR STAFF DEVOPS ENGINEER
Automated Testing at Santa Monica Studio ♦ Tools Summit
MONDAY, MARCH 20 ♦ 4:40PM – 5:10PM ♦ ROOM 3004, WEST HALL

XUANYI ZHOU ♦ PROGRAMMER
Real-time Neural Texture Upsampling in 'God of War Ragnarök' ♦ Machine Learning Summit
TUESDAY, MARCH 21 ♦ 2:10PM – 2:40PM ♦ ROOM 2010, WEST HALL

ETHAN AYER ♦ SR ENVIRONMENT ARTIST
The Art of Making Vistas ♦ Art Summit
TUESDAY, MARCH 21 ♦ 3:00PM – 3:30PM ♦ ROOM 3007, WEST HALL

GÖKSU UĞUR ♦ AI LEAD
Preparing AI Systems For God of War Ragnarök ♦ Programming
WEDNESDAY, MARCH 22 ♦ 9:00AM – 10:00AM ♦ ROOM 303, SOUTH HALL

VICKI SMITH ♦ SR STAFF LEVEL DESIGNER
The Final Battle of 'God of War Ragnarök': Techniques For Delivering High-stakes Sequences ♦ Design
WEDNESDAY, MARCH 22 ♦ 10:30AM – 11:00AM ♦ ROOM 2002, WEST HALL

STEPHEN McAULEY ♦ LEAD RENDERING PROGRAMMER
Rendering 'God of War Ragnarök' ♦ Programming
WEDNESDAY, MARCH 22 ♦ 2:00PM – 3:00PM ♦ ROOM 303, SOUTH HALL

ERIC GOTTESMAN ♦ SR STAFF DEVOPS ENGINEER
Modernizing multiplayer services for "God of War: Ascension" (PS3) ♦ Production & Team Leadership ♦ Presented by Amazon Web Services
WEDNESDAY, MARCH 22 ♦ 2:00PM – 3:00PM ♦ GDC PARTNER STAGE, EXPO FLOOR, NORTH HALL

SAM STERNKLAR ♦ SR PROGRAMMER
'God Of War Ragnarök's' Visual Scripting Solution ♦ Programming
THURSDAY, MARCH 23 ♦ 10:00AM – 11:00AM ♦ ROOM 2006, WEST HALL

ADAM OLIVER ♦ SR COMBAT DESIGNER
Breaking Barriers: Combat Accessibility in 'God of War Ragnarök' ♦ Design
THURSDAY, MARCH 23 ♦ 2:00PM – 2:30PM ♦ ROOM 2002, WEST HALL

GÖKSU UĞUR ♦ AI LEAD
Practical Tools for Transitioning Into Leadership Roles ♦ Leadership
THURSDAY, MARCH 23 ♦ 2:00PM – 2:30PM ♦ ROOM 303, SOUTH HALL

ZACH BOHN ♦ SR STAFF TECHNICAL UI DESIGNER
'God of War Ragnarök': Building The UI For a AAA Title ♦ Design
THURSDAY, MARCH 23 ♦ 4:00PM – 5:00PM ♦ ROOM 303, SOUTH HALL

SALAAR KOHARI ♦ PROGRAMMER
Companion Traversal in 'God of War Ragnarök'♦ Programming
FRIDAY, MARCH 24 ♦ 10:00AM – 11:00AM ♦ ROOM 2002, WEST HALL

TENGHAO WANG ♦ SR PROGRAMMER
Joint-based Skin Deformation in 'God of War Ragnarök'♦ Programming
FRIDAY, MARCH 24 ♦ 1:30PM – 2:30PM ♦ ROOM 2006, WEST HALL

HARLEIGH AWNER ♦ TECHNICAL NARRATIVE DESIGNER
How to Build a Home: Designing Narrative For Sindri's House in 'God of War Ragnarök'♦ Design
FRIDAY, MARCH 24 ♦ 3:00PM – 3:30PM ♦ ROOM 2001, WEST HALL

Santa Monica Studio GDC

Santa Monica Studio

Our journey
Your story

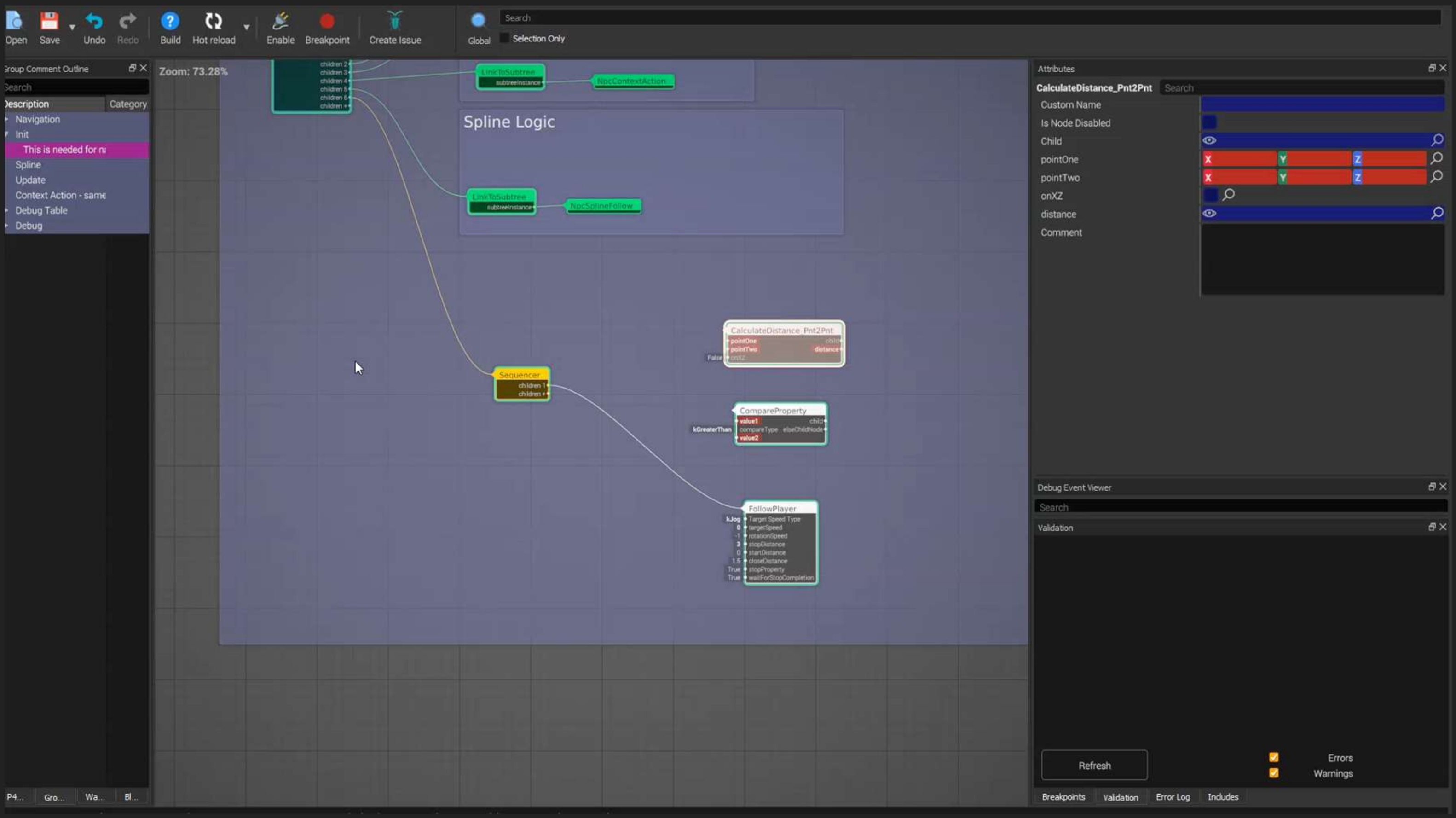
We're hiring for what's next!

We're expanding our family across disciplines and would love to meet you. Please visit sms.playstation.com/careers for all openings or drop us a line at sms.recruiting@sony.com

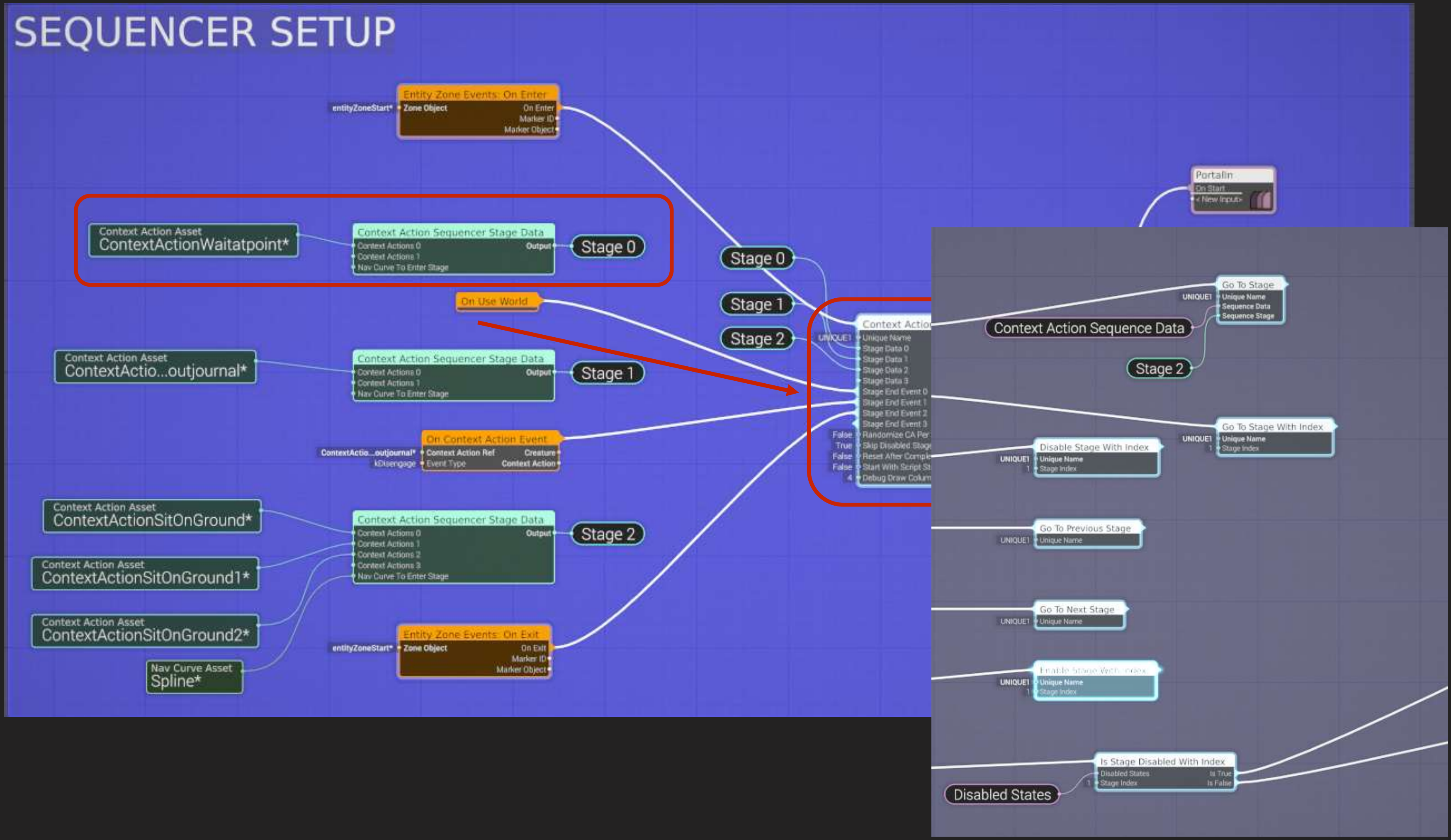
@ santamonicastudio @ SonySantaMonica @ santamonicastudio

Extra slides

Santa Monica Studio



Some more quality of life improvement features of behavior tree editor



We even had an additional "Context Action Sequencer" module that allowed us to easily mix and match and create more complicated setups with multiple context actions while minimizing the need for each
module that allows designers to an easy setup for sequencing.
With this module designers can create "Context Action Sequencer Stage Data" that allowed them to pass in context actions per stage and manage the trigger for each stage to be handled by the module automatically. They also had different options in terms of how sequencer operates and callbacks when certain things happen to tie events in level if necessary
On top of that we had API calls like you see in this image. It also threw certain events with useful information at the time for designers to hook things up to like when a stage is completed
This was a common setup for cases like puzzles in levels that required a sequence of actions (put a gif that shows this working in a level) as well as narrative spaces where AI can walk around in the space doing multiple actions (Freya Garden)